

On the implausibility of classical client blind quantum computing

Scott Aaronson^{*1}, Alexandru Cojocaru^{†2}, Alexandru Gheorghiu^{‡2}, and Elham Kashefi^{§2,3}

¹*Department of Computer Science, University of Texas at Austin*

²*School of Informatics, University of Edinburgh*

³*CNRS LIP6, Université Pierre et Marie Curie, Paris*

Abstract

Suppose a large scale quantum computer becomes available over the Internet. Could we delegate universal quantum computations to this server, using only classical communication between client and server, in a way that is information-theoretically blind (i.e., the server learns nothing about the input apart from its size, with no cryptographic assumptions required)? In this paper we give strong indications that the answer is no. This contrasts with the situation where quantum communication between client and server is allowed — where we now know, from work over the past decade, that such information-theoretically blind quantum computation is possible. It also contrasts with the case where cryptographic assumptions are allowed: there again, it is now known that there are quantum analogues of fully homomorphic encryption.

In more detail, we observe that, if there exist information-theoretically secure classical schemes for performing universal quantum computations on encrypted data, then we get unlikely containments between complexity classes, such as $\text{BQP} \subset \text{NP/poly}$. Moreover, we prove that having such schemes for delegating quantum sampling problems, such as BOSONSAMPLING , would lead to a collapse of the polynomial hierarchy. We then consider encryption schemes which allow one round of quantum communication and polynomially many rounds of classical communication, yielding a generalization of blind quantum computation. We give a complexity theoretic upper bound, namely $\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$, on the types of functions that admit such a scheme. This upper bound then lets us show that, under plausible complexity assumptions, such a protocol is no more useful than classical schemes for delegating NP -hard problems to the server. Lastly, we comment on the implications of these results for the prospect of verifying a quantum computation through classical interaction with the server. We argue that if such a procedure is possible, then it must either reveal much of the computation to the quantum computer or rely on computational assumptions.

1 Introduction

An important area of research in modern cryptography is that of performing *computations on encrypted data*. The general idea is that a client wants to compute some function f on some input x , but lacks the computational power to do this in a reasonable amount of time. Luckily, the client has access to a computationally powerful server (cloud, cluster etc) which can compute $f(x)$ quickly. However, due to the nature of the computation (which might involve sensitive or classified information), or the level of security of the server (which could be compromised remotely), we would like that the input x is hidden from the server at all times. The client can simply encrypt x , but then the question arises: how should the server act on this encrypted input so that the client can, in the end, recover $f(x)$? The general problem of computing on encrypted data was first considered by Rivest, Adleman and Dertouzos [1]. Since then, instances of this problem have

^{*}Email: aaronson@cs.utexas.edu

[†]Email: a.cojocaru@sms.ed.ac.uk

[‡]Email: a.gheorghiu@sms.ed.ac.uk

[§]Email: ekashefi@inf.ed.ac.uk

become commonplace, especially when considering applications such as electronic voting, machine learning on encrypted data, program obfuscation and others [2–7].

This problem, at least for the case of classical computations, was addressed in 2009 by the breakthrough result of Gentry in which he introduced a scheme for *fully homomorphic encryption* [8]. In homomorphic encryption the client has efficient algorithms for encryption Enc , and decryption Dec , satisfying the property that $Dec(f, x, Eval(f, Enc(x))) = f(x)$, for any function f from some set \mathcal{C} . In other words, the server computes some evaluation of f on the encrypted input $Enc(x)$ (using $Eval$) and returns this to the client which can then decrypt it to $f(x)$. Of course, the server should not be able to infer information about x from $Enc(x)$, a condition which is typically expressed through *semantic security* [9]. The scheme is therefore secure under suitable *cryptographic assumptions*. If the set \mathcal{C} contains all polynomial-sized circuits then the scheme becomes a fully homomorphic encryption scheme, commonly abbreviated FHE.

Computing on encrypted data becomes particularly interesting when considering the server to be a *quantum computer*. This is because efficient quantum algorithms have been found for various problems which are believed to be intractable for classical computers. In fact, if one is only given black-box or oracle access to certain functions then an *exponential separation* between quantum computation and classical probabilistic computation is provable [10–13]. Classical clients would therefore be highly motivated to delegate problems to quantum computers. However, ensuring the privacy of the inputs to these problems, under cryptographic assumptions, is challenging. In particular, we’re faced with the following challenges:

- Devise an encryption scheme which is secure against quantum computers and does not leak information to the server about the client’s input.
- Ensure that the encryption scheme allows the client to recover the output of the computation from the result provided by the quantum server.
- The whole protocol must be efficient for the client. Ideally, the number of rounds of interaction between the client and the server, as well as the client’s local computations should scale, at most, polynomially with the size of the input.

Surprisingly, in spite of these stringent requirements, such a scheme already exists and is known as *universal blind quantum computation* or UBQC [14, 15].

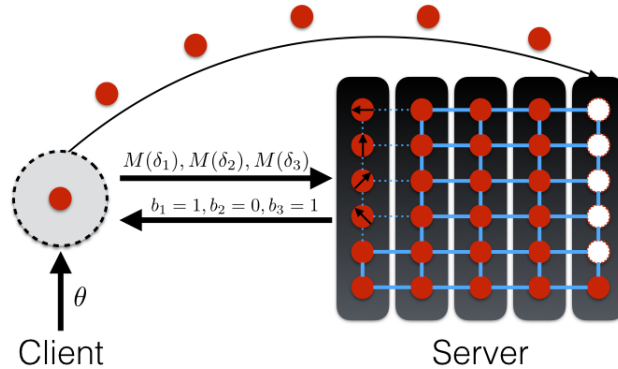


Figure 1: Universal Blind Quantum Computation

In UBQC, a classical client augmented with the ability to prepare single qubits sends these qubits to the server along with instructions on how to entangle and measure them in order to perform a computation. Because of fundamental limitations imposed by quantum mechanics (such as the inability to clone quantum information), it is possible for the server to perform any general quantum computation without learning anything about it except for an upper bound on its size. In other words, UBQC provides *information-theoretically* secure quantum function evaluation with a minimalistic quantum client. An earlier protocol by Childs [16] achieved the same functionality,

but had the client perform more complicated quantum operations. An illustration of the protocol, adapted from [17], is given in Figure 1.

UBQC is one of the foundations of the field of quantum computing on encrypted data and has been improved, extended and used in various other works [18–25]. Other approaches to blind quantum computing also exist such as the previously mentioned protocol of Childs and a more recent protocol of Broadbent [26] that has also led to a number of applications [27, 28]. However, the limiting factor of all these approaches is that the client needs to prepare and send single qubits. We would like to know if this requirement can be removed and so we pose the question:

Is there a scheme for blind quantum computing that is information-theoretically secure, and that requires only classical communication between client and server?

More precisely, the question is whether it is possible for a classical client running in polynomial time to obtain the value $f(x)$ through classical interactions with a quantum server, such that $f(x)$ can be computed by a quantum computer in time polynomial in the length of x (i.e., $f \in \text{BQP}$). Additionally, the number of rounds of interaction should be polynomial in the length of x and at the end of the protocol the server should learn, at most, the length of x and nothing more, under no cryptographic assumptions.

In this paper we provide compelling evidence that the answer is no, this is not possible: not all of these conditions can be met at the same time. Importantly, our result does not contradict the existence of FHE, based on cryptographic assumptions: we are interested only in information-theoretic security.

In addition to this result, we also provide a complexity-theoretic upper bound for the types of functions which can be evaluated by UBQC-type protocols, in which the client has some quantum capabilities. We show that, under plausible complexity assumptions, this upper bound prohibits the client from delegating NP-hard functions to the server.

1.1 Organization

Our results are centred around the concept of a *generalised encryption scheme* (GES) introduced by Abadi, Feigenbaum and Killian [29] which is formally defined Section 2. Roughly speaking, a GES is a protocol between a probabilistic polynomial-time *classical* client (BPP) and a computationally unbounded server for computing on encrypted data. The client sends the server a description of some function f , to be evaluated. Using some polynomial-time algorithm denoted E , the client encrypts its input x , and sends $E(x)$ to the server. The server and the client then interact for a number of rounds which is polynomial in the size of x . Finally, using a polynomial-time decryption algorithm, denoted D , the client “decrypts” the server’s responses and obtains $f(x)$ with probability $1/2 + 1/\text{poly}(|x|)$. Importantly, throughout the protocol the server learns, at most, the size of x . Having the server be computationally unbounded means the scheme requires information-theoretic security. Abadi et al. gave a complexity theoretic upper bound on the types of functions that admit such a scheme. They showed that any function f with a GES protocol is contained in the class $\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$. In our work, we explore the implications of having a GES for different types of quantum computations and show that this leads to highly unlikely containments of complexity classes.¹

Our study of generalised encryption schemes for quantum computations begins in Section 3. We start by looking at decision problems. Specifically we consider polynomial-time quantum computations, BQP. From the Abadi et al. result, it immediately follows that having a GES for all BQP functions implies that $\text{BQP} \subset \text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$. This containment seems unlikely and while we cannot reduce it to something more standard, such as a collapse of the polynomial hierarchy, we provide oracle evidence in support of the conjecture $\text{BQP} \not\subset \text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$. In particular we show that there exists an oracle O such that $\text{BQP}^O \not\subset (\text{NP}/\text{O}(n^d))^O$.

¹Definitions of complexity classes used in this paper can be found in the Complexity Zoo [30]. In Appendix 7.1, we provide a brief overview of advice, oracles and sampling classes.

To add more weight to our conjecture, we then consider the implications of having a GES for sampling problems. In particular, we consider *BosonSampling*, a problem which is efficiently solvable by a quantum computer but believed to be intractable for classical probabilistic computers running in polynomial time [31]. We first redefine generalised encryption schemes for sampling problems, and then show that having such a scheme for exact BosonSampling implies a collapse of the polynomial hierarchy. A similar result holds for approximate BosonSampling, assuming a conjecture by Aaronson and Arkhipov, regarding estimating the permanent of a Gaussian matrix, holds [31].

Next, in Section 4 we define a quantum GES (QGES) which can be viewed as a generalization of UBQC, in the sense that the client can send one quantum message to the server. While our previous results seem to rule out GES protocols for BQP computations, we know that giving the client some minimal quantum capabilities removes this limitation. In the spirit of the Abadi et al. result, it is then natural to consider QGES protocols, in which the client is no longer classical, and investigate the complexity-theoretic upper bounds of functions which admit such a protocol, also leaking at most the size of the input. We find that for QGES protocols, having an extra property known as *offline-ness*, such functions are contained in the class $\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$. Roughly speaking, an offline protocol is one in which the client does not need to commit to any particular input (of a given size), after having sent the first encrypted message to the server. In other words, there is some efficient operation which the client can use to change its input after having initiated communication with the server. We also show that if $\text{NP} \subset \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$, then a result similar to the collapse of the polynomial hierarchy is true: namely, $\Pi_3^P \subseteq \text{NP}^{\text{NP}^{\text{PromiseQMA}}}$. Hence, while QGES does (by definition) allow for delegating BQP computations, it seems to be no more useful than the regular GES at delegating NP-hard functions.

Lastly, in Section 5 we discuss the implications of our results for quantum verification. Specifically, we emphasize that many existing protocols for verifying general quantum computations rely on blindness in the sense of hiding the input from the quantum server, in an information-theoretic sense, apart from an upper bound on its size [21, 27, 32]. Our results render it unlikely that there exist classical verification schemes that also rely on this type of blindness.

The motivation for our results is twofold. Firstly, the complexity-theoretic approach we use lets us determine very precisely what is and what isn't possible with quantum computing on encrypted data. In particular, our “no-go” result for classical clients informs the direction of future research in this field: we either have to consider protocols leaking more information to the server, or consider schemes with computational security.

Secondly, we emphasize the significance of the complexity theoretic upper bound on functions which can be computed with a QGES, as well as the result that NP-hard functions are unlikely to satisfy this bound. Quantum computers could, in principle, solve NP-complete problems quadratically faster than classical computers, thanks to Grover's algorithm [33]. Even though the speedup of Grover's algorithm is only quadratic, from (say) 2^n to $2^{n/2}$, our no-go theorem is only concerned with the length of the computation performed on the client side, and therefore applies to Grover's algorithm just as it would to a quantum algorithm achieving exponential speedup. Our result shows that clients cannot exploit the Grover speedup, even when allowing some quantum communication, if we also want to keep their inputs hidden in an information-theoretic sense.

1.2 Related work

As mentioned, the problem of computing on encrypted data was first considered by Rivest, Adleman and Dertouzos [1], which then led to the development of *homomorphic encryption* and eventually to fully homomorphic encryption with Gentry's scheme [8]. Since then there have been many other FHE protocols relying on more standard cryptographic assumptions and having more practical requirements [34–36].

While FHE is similar to GES in many respects, there are also significant differences. For starters, FHE protocols have only one round of interaction between the client and the server, whereas GES allows for polynomially many rounds. Additionally, GES assumes the server is

computationally unbounded and hence requires information-theoretic security. In contrast, FHE relies on computational security. More precisely FHE schemes have semantic security against polynomial-time (quantum) algorithms [8].

The problem of *quantum* computing on encrypted data was introduced by Childs [16] and Arrighi and Salvail [37]. Further development eventually led to UBQC [14, 15] and the scheme of Broadbent [26]. The latter was followed by the construction of the first schemes for quantum fully homomorphic encryption (QFHE) [28, 38]. For a review of blind quantum computing and related protocols see [39].

In existing QFHE schemes, the server is a BQP machine and the client has some quantum capabilities of its own, although it is not able to perform universal quantum computations. Both the size of the exchanged messages and the number of operations of the client are polynomial in the size of the input. Similar to FHE, these protocols rely on computational assumptions for security [40] and involve one round of back and forth interaction between the client and the server [28, 38]. QFHE with information-theoretic security (and a computationally unbounded server) has been considered by Yu et al. in [41], where it is shown that it is impossible to have such a scheme for arbitrary unitary operations (or even arbitrary reversible classical computations). They do not, however, consider an upper bound on the types of Boolean functions that can be computed in that setting. Importantly, their result does not rule out an information-theoretic secure QFHE protocol for polynomial-sized quantum computations. In relation to our work, QFHE with information-theoretic security can be viewed as a one-round QGES in which the server responds with a quantum message. The complexity upper bound we prove for QGES computable functions would then apply for QFHE as well (provided that in QFHE we only leak the size of the input to the server), since our proof allows a quantum message from the server just as well as a classical message.

The possibility of a classical client delegating a blind computation to a quantum server was considered by Morimae and Koshiba [42]. They showed that such a protocol in which the client leaks no information about its input to the server and there is only one round of interaction leads to $\text{BQP} \subseteq \text{NP}$, considered an unlikely containment. We consider the more general setting of a GES for BQP functions, where the number of rounds can be polynomial in the size of the input and we allow the encryption to leak the size of the input. In fact, the question of whether a GES, as defined in Abadi et al., can exist for quantum computations was raised before by Dunjko and Kashefi [43].

Since our result is primarily about the implausibility of having a GES for BQP computations, there are a number of caveats which, if circumvented, could still lead to a reasonable scheme for classically delegating computations to a quantum server. These are:

- The requirement that the client leaks at most the size of the input to the server. If one is allowed to leak more information then our result no longer applies. Indeed, such a scheme already exists. Mantri et al. proposed a protocol relying on the measurement-based model of quantum computation in which a client can delegate a quantum computation to a server while still retaining some privacy [44].
- The requirement of information-theoretic security. As mentioned, the QFHE schemes, while not having a fully classical client, are based on computational security. Moreover, for a subclass of quantum computations called instantaneous quantum computations (IQP), Shepherd and Bremner proposed a computationally secure scheme for delegating encrypted IQP problems to a quantum server [45].
- The requirement that the client interacts with only one quantum server. A number of schemes have been proposed where a classical client can delegate quantum computations to multiple servers as long as these servers share entanglement and are not allowed to communicate during the protocol [32, 46, 47].

2 Cryptographic preliminaries

The basis of most of the results in our paper is the generalised encryption scheme. We state its definition from [29]:

Definition 1 ([29] Generalised Encryption Scheme (GES)). *A generalised encryption scheme (GES) is a two party protocol between a classical client C , and an unbounded server S , characterized by:*

- *A function² $f : \Delta \rightarrow \Sigma$, where $\Delta, \Sigma \subseteq \{0, 1\}^*$.*
- *A cleartext input $x \in \text{Domain}(f)$, for which the client wants to compute $f(x)$.*
- *An expected polynomial-time key generation algorithm K which works as follows: for any $x \in \text{Domain}(f)$, with probability greater than $1/2 + 1/\text{poly}(|x|)$ we have $(k, \text{success}) \leftarrow K(x)$, where $k \in \{0, 1\}^{\text{poly}(|x|)}$. If the algorithm does not return success then we have $(k', \text{fail}) \leftarrow K(x)$, where $k' \in \{0, 1\}^{\text{poly}(|x|)}$.*
- *A polynomial-time deterministic algorithm E which works as follows: for any $x \in \text{Domain}(f)$, $k \in \{0, 1\}^{\text{poly}(|x|)}$ and $s \in \{0, 1\}^{\text{poly}(|x|)}$ we have that $y \leftarrow E(x, k, s)$, where $y \in \{0, 1\}^{\text{poly}(|x|)}$.*
- *A polynomial-time deterministic decryption algorithm D , which works as follows: for any $x \in \text{Domain}(f)$, $k \in \{0, 1\}^{\text{poly}(|x|)}$ and $s \in \{0, 1\}^{\text{poly}(|x|)}$ we have that $z \leftarrow D(s, k, x)$, where $z \in \{0, 1\}^{\text{poly}(|x|)}$.*

And satisfying the following properties:

1. *There are m rounds of communication, such that $m = \text{poly}(|x|)$. Denote the client's message in round i as c_i and the server's message as s_i .*
2. *On cleartext input x , C runs the key generation algorithm until success to compute a key $(k, \text{success}) = K(x)$. This happens before the communication between C and S is initiated, and the key k is used throughout the protocol.*
3. *In round i of the protocol, C computes $c_i = E(x, k, \bar{s}_{i-1})$, where \bar{s}_{i-1} denotes the server's responses up to and including round $i-1$, i.e. $\langle s_0, s_1 \dots s_{i-1} \rangle$. We assume that s_0 is the empty string. C then sends c_i to S .*
4. *In round i of the protocol, S responds with s_i , such that $s_i \in \{0, 1\}^{\text{poly}(|x|)}$. Additionally, the server's responses are drawn probabilistically from a distribution which is consistent with property 5.*
5. *At the end of the protocol, C computes $z \leftarrow D(\bar{s}_m, k, x)$ and with probability $1/2 + 1/\text{poly}(|x|)$, we have that $z = f(x)$.*

Let us provide some intuition for this definition. The purpose of a GES is to allow a client to compute some $f(x)$ which it cannot compute with its own resources. It does this by interacting with a computationally powerful server for a number of rounds which is polynomial in the size of the input. Importantly, the GES allows the client to hide some information about x from the server. We make this statement more precise through the following definition:

Definition 2. *An encryption algorithm E for some function f with a priori distributions over the input denoted by the random variable X , leaks at most $L(X)$ if for all X , and for all $l \in \text{Range}(L)$, the random variables X and $f(X)$ are independent given $l \leftarrow L(X)$.*

²Note that f need not be a total function since it is defined on a subset of $\{0, 1\}^*$. Whenever we say that $f \in \mathcal{C}$, for some complexity class \mathcal{C} , we mean that there exists a total function $g \in \mathcal{C}$ such that $g(x) = f(x)$, for all $x \in \text{Domain}(f)$.

Finally, we state the main theorem from [29], which we will use throughout the paper:

Theorem 1 ([29] GES leaking size of input). *If a function f admits a GES which leaks at most the size of the input (i.e. $L(X) = |X|$), then $f \in \text{NP/poly} \cap \text{coNP/poly}$ ³.*

We include a simplified proof of this theorem in Appendix 7.2. It should be mentioned that if the client is a BPP machine and the functions computable with the GES are contained in $\text{NP/poly} \cap \text{coNP/poly}$ then we should in fact be working with the class $\text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}}$. However, it is not very difficult to show that $\text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}} = \text{NP/poly} \cap \text{coNP/poly}$. This can be done using a result of Brassard [48] which shows that $\text{P}^{\text{NP} \cap \text{coNP}} = \text{NP} \cap \text{coNP}$, together with Adleman's theorem (that $\text{BPP} \subset \text{P/poly}$) [49]. The proof of this fact can also be found in Appendix 7.2.

When working with generalised encryption schemes we will be interested in delegating BQP-complete problems since these capture the power of polynomial-time quantum computations⁴. We therefore state the definition of the canonical BQP-complete problem:

Definition 3 ([27, 28] Q-CIRCUIT). *The problem Q-CIRCUIT consists of a quantum circuit made of a sequence of gates, $U = U_N, \dots, U_1$ acting on an n -qubit input $|0^n\rangle$ (we take these circuits to be given in the universal gateset $\text{GATES} = \{\text{X}, \text{Z}, \text{H}, \text{CNOT}, \text{T}\}$). Let $p(U) = \|\lvert 0 \rangle \langle 0 \rvert \otimes \mathbb{I}_{n-1} U \lvert 0^n \rangle\|^2$ be the probability of observing “0” as a results of a computational basis measurement of the n^{th} output qubit, obtained by evaluating U on input $|0^n\rangle$.*

The input to the problem (not to be confused with the input to the circuit which is the state $|0^n\rangle$) is denoted x and consists of the description of the quantum circuit U . The problem is then to decide whether:

$$\begin{aligned} x \in \text{Q-CIRCUIT}_{\text{YES}} : p(U) &\geq 2/3 \\ x \in \text{Q-CIRCUIT}_{\text{NO}} : p(U) &\leq 1/3 \end{aligned}$$

promised that one of these is the case.

We will now give the definition of Universal Blind Quantum Computation (UBQC) adapted from [14]. Note that the definition relies on elements of *measurement-based quantum computation* [50, 51] for which we give a brief introduction in Section 7.1.2.

Definition 4 ([14] Universal Blind Quantum Computation (UBQC)). *A Universal Blind Quantum Computation (UBQC) scheme for input x (resulting from a random variable X) is a two party protocol between a client C , and a server S which allows the client to decide the Q-CIRCUIT problem on input x (corresponding to some quantum circuit \mathcal{C}), while leaking at most the size of x to the server. We assume that the circuit \mathcal{C} which is used to perform the Q-CIRCUIT problem has N quantum gates. The protocol then works as follows:*

- *C prepares $M = O(N)$ qubits initialized in the states $|+\theta_i\rangle$, with i going from 1 up to M . Each angle θ_i is chosen uniformly at random from the set $\{0, \pi/4, 2\pi/4, \dots, 7\pi/4\}$.*
- *C sends the M qubits to S .*
- *S entangles the qubits, using CZ operations, according to a graph structure allowing universal quantum computations (see Section 7.1.2 for an example).*
- *C instructs S to measure each qubit, i , at the angles $\delta_i = \theta_i + (-1)_i^s \phi_i + r_i \pi$. The angles ϕ_i are the computation angles, corresponding to the computation of the circuit \mathcal{C} . Note that $\phi_i \in \{0, \pi/4, 2\pi/4, \dots, 7\pi/4\}$. The terms $s_i, r_i \in \{0, 1\}$ specify a correction to be applied on the current qubit, based on previous measurement outcomes.*

³Note that from the definition of f , $\text{Range}(f)$ is not necessarily $\{0, 1\}$, corresponding to a decision problem, but can be any subset of $\{0, 1\}^*$. In this case the correct containment would be $f \in \text{FNP/poly} \cap \text{FcoNP/poly}$, where FNP and FcoNP are the relational versions of NP and coNP. This distinction is not crucial for understanding the result and we can always simply restrict to decision problems.

⁴Strictly speaking, no BQP-complete problems are known and we are instead referring to PromiseBQP-complete problems. These are problems which have a certain promise on the input and are solvable by a polynomial-time quantum algorithm. The Q-CIRCUIT problem, defined above, is a PromiseBQP-complete problem.

- S responds with the appropriate measurement outcomes.

The defining feature of UBQC is *blindness* for which we also provide the definition from [14]:

Definition 5. Let P be a UBQC protocol with input drawn from the random variable X and let $L(X)$ be any function of this random variable. We say that the protocol is blind while leaking at most $L(X)$ if, on the client's input X , for any $l \in \text{Range}(L)$, the following two hold when given $l \leftarrow L(X)$:

1. The distribution of the classical information obtained by the server in P is independent of X .
2. Given the distribution of classical information described in 1, the state of the quantum system obtained by the server in P is fixed and independent of X .

Note the similarities between UBQC and GES:

- Both are client-server protocols for allowing the client to compute some hard function while hiding information from the server.
- Both provide information-theoretic security.
- The case of interest for GES is when the scheme leaks at most the size of the input to the server. In UBQC, already the scheme leaks at most the size of the input to the server.

It thus seems that UBQC is a particular instance of a type of quantum GES (a GES with a quantum client). We will elaborate more on this in Section 4. As a final remark, notice that Q-CIRCUIT is a problem in which the input is the description of the quantum computation (circuit) itself. Therefore, UBQC keeps the description of this circuit private from the server in an information-theoretic way. This is similar to uses of FHE in which both the data and the algorithm that should process the data are sent in encrypted form to the server [52].

3 GES for quantum problems

We would like to investigate the possibility of having a GES for efficient quantum problems. Recall that the GES entails a BPP client delegating problems to an unbounded server. We will be examining both the cases of a GES for delegating BQP decision problems, as well as the case for problems like BOSONSAMPLING and SampBQP in general. We show that in both instances, having schemes which leak at most the size of the input lead to very unlikely consequences. In effect, this provides evidence for the implausibility of classical client blind quantum computation with information-theoretic security. In Section 5, we will also comment on the implications of this fact for classical client quantum verification.

As mentioned, the result of Abadi et al. shows that functions which admit a GES are contained in the complexity class $\text{NP/poly} \cap \text{coNP/poly}$. In fact they use this result to argue that any class of problems which contains NP is not suitable for delegation through a GES, since if $\text{NP} \subset \text{NP/poly} \cap \text{coNP/poly}$, then, in particular, $\text{NP} \subset \text{coNP/poly}$. Using a result of Yap [53], that if $\text{NP} \subset \text{coNP/poly}$ then $\Sigma_3^P \subseteq \Pi_3^P$ and the polynomial hierarchy collapses at the third level.

In our case, the class of interest is BQP and so, delegating efficient quantum computations using a GES reduces to asking whether the following containment holds: $\text{BQP} \subset \text{NP/poly} \cap \text{coNP/poly}$. Determining the truthfulness of this containment is no more easier than determining whether $\text{P} = \text{NP}$, so we cannot give a definite proof either way. However, we can provide reasonable evidence that the containment does not hold. Ideally, we would like to construct an oracle O such that $\text{BQP}^O \not\subset (\text{NP/poly} \cap \text{coNP/poly})^O$. Unfortunately, this is not easy to do either as having such an oracle would also separate BQP^O from AM^O , a problem which has eluded complexity theorists for some time [54]. However, if we fix the degree of the polynomial determining the size of the advice, we can prove the following:

Theorem 2. *For each $d \in \mathbb{N}$, there exists an oracle O_d , such that BQP^{O_d} is not contained in $(\text{NP}/\text{O}(n^d))^{O_d}$.*

The complete proof can be found in Appendix 7.3.

One can argue that oracle results do not constitute compelling evidence regarding the relationship between complexity classes. Indeed, it has been known for a while that there exist oracles such that $\text{P}^O \neq \text{NP}^O$ and oracles such that $\text{P}^O = \text{NP}^O$. Moreover there are non-relativizing results such as $\text{IP} = \text{PSPACE}$ even though there exists an oracle such that $\text{IP}^O \neq \text{PSPACE}^O$. However, oracles allow us to study the query complexity of problems in different models of computation. In fact there are situations in practice where computer programs are restricted to making black-box calls to functions in order to determine their properties [55]. Apart from this, past oracle results have proven to be insightful for the development of algorithms and complexity theory. Most notably Simon's oracle separation between BPP and BQP led to Shor's algorithm for factoring and computing discrete logarithm [56]. More arguments in defence of oracles are provided in Section 1.3 of [54].

As we have seen, providing strong evidence in the case of decision problems is difficult. The situation is somewhat better in the case of sampling problems, thanks to results such as the Aaronson-Arkhipov result on BosonSampling and the Bremner, Josza and Shepherd result on instantaneous quantum computation [31, 57]. In this case, the problem we consider is that of having a GES for sampling problems. Concretely, the client has as input a string x , which specifies a distribution \mathcal{D}_x , and would like to use the GES to obtain a sample from that distribution. For example, x could specify a linear optics network and the distribution can represent the probability of observing certain numbers of photons per each mode of the network. This is the BosonSampling problem, denoted BOSONSAMPLING for which we provide the definition in Section 7.1.3 where we also give the definition of sampling problems. In this case the server should learn at most the size of the linear optics network. When we are interested in approximate sampling, we can assume that the ε from Definition 9 (the variation distance between the distribution that we would like to sample from and the one that is actually sampled) is fixed in advance and known to both the client and the server.

The previously mentioned results show that classical randomized algorithms cannot sample efficiently from certain distributions, unless the polynomial hierarchy collapses. Importantly, those distributions can be efficiently sampled by a quantum computer. Using these results we can prove the following:

Theorem 3. *There does not exist a GES for exact BOSONSAMPLING, leaking at most the size of the input, unless PH collapses at the 4th level.*

For this theorem we are assuming that the server samples exactly not approximately and hence the GES allows the client to obtain samples from the distribution defined in the BOSONSAMPLING problem (as opposed to a distribution which is close to it). Formally, we need to first define the notion of GES for sampling problems, since the definition we have is for deterministic functions.

Definition 6 (Generalised Encryption Scheme for Sampling Problems). *A generalised encryption scheme for sampling problems is a two party protocol between a classical client C , and an unbounded server S , characterized by:*

- *A collection of probability distributions $(\mathcal{D}_x)_{x \in \{0,1\}^*}$, where \mathcal{D}_x is a distribution over $\{0,1\}^{\text{poly}(|x|)}$.*
- *A cleartext input $x \in \{0,1\}^*$, specifying a particular distribution \mathcal{D}_x .*
- *An expected polynomial-time key generation algorithm K which works as follows: for any $x \in \{0,1\}^*$, with probability greater than $1/2 + 1/\text{poly}(|x|)$ we have $(k, \text{success}) \leftarrow K(x)$, where $k \in \{0,1\}^{\text{poly}(|x|)}$. If the algorithm does not return success then we have $(k', \text{fail}) \leftarrow K(x)$, where $k' \in \{0,1\}^{\text{poly}(|x|)}$.*

- A polynomial-time deterministic algorithm E which works as follows: for any $x \in \{0,1\}^*$, $k \in \{0,1\}^{\text{poly}(|x|)}$ and $s \in \{0,1\}^{\text{poly}(|x|)}$ we have that $y \leftarrow E(x, k, s)$, where $y \in \{0,1\}^{\text{poly}(|x|)}$.
- A polynomial-time deterministic decryption algorithm D , which works as follows: for any $x \in \{0,1\}^*$, $k \in \{0,1\}^{\text{poly}(|x|)}$ and $s \in \{0,1\}^{\text{poly}(|x|)}$ we have that $z \leftarrow D(s, k, x)$, where $z \in \{0,1\}^{\text{poly}(|x|)}$.
- A closeness bound $\varepsilon \geq 0$ specifying the variation distance between the ideal distribution \mathcal{D}_x that the client would like to sample from and the one obtained in the protocol.

And satisfying the following properties:

1. There are m rounds of communication, such that $m = \text{poly}(|x|)$. Denote the client's message in round i as c_i and the server's message as s_i .
2. On cleartext input x , C runs the key generation algorithm until success to compute a key $(k, \text{success}) = K(x)$. This happens before the communication between C and S is initiated, and the key k is used throughout the protocol.
3. In round i of the protocol, C computes $c_i = E(x, k, \bar{s}_{i-1})$, where \bar{s}_{i-1} denotes the server's responses up to and including round $i-1$, i.e. $\langle s_0, s_1 \dots s_{i-1} \rangle$. We assume that s_0 is the empty string. C then sends c_i to S .
4. In round i of the protocol, S responds with s_i , such that $s_i \in \{0,1\}^{\text{poly}(|x|)}$. Additionally, the server's responses are drawn probabilistically from a distribution which is consistent with property 5.
5. At the end of the protocol, C computes $z \leftarrow D(\bar{s}_m, k, x)$ and we have that z is a sample from some distribution \mathcal{C}_x such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$.

With this definition it is clear that in Theorem 3 the collection of distributions are those defined by BOSONSAMPLING and the closeness bound is taken to be zero ($\varepsilon = 0$). Importantly, unlike the definition for decision problems there is no requirement that the “correct output” is obtained with high probability since in this case there is no “correct output” to some deterministic function. Instead, the equivalent notion to “obtaining the correct outcome with high probability” is asking that the variation distance ε be small. For BOSONSAMPLING, having a GES entails the following:

- The client's input x is essentially the column-orthonormal matrix A from Definition 10 which specifies a linear optics network and hence a distribution \mathcal{D}_x . This input is encrypted and sent to the server.
- The server has the ability to “query” the BOSONSAMPLING oracle \mathcal{O} from Definition 10. The oracle receives as input a description of the linear optics network and a random string and returns a sample from the distribution obtained by measuring the number of photons per mode in the network. We say “query” because if the server is a quantum computer then it does not query some deterministic function \mathcal{O} with an input and random string, since the randomness is intrinsic to the computation. However, since the client only specifies the input distribution (in encrypted form) one can model the server as if it were querying the deterministic oracle \mathcal{O} with external randomness.
- After a number of rounds of interaction which is polynomial in the size of the optical network the client obtains a sample from the distribution \mathcal{D}_x .

The proof of Theorem 3 relies on showing that if such a GES existed then the BOSONSAMPLING oracle \mathcal{O} could be simulated in $\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$. Combining this with the result of Theorem 7 and other known relations between complexity classes leads to a collapse of the polynomial hierarchy. The proof can be found in Appendix 7.4.

In practice, however, if we want the server to solve a sampling problem for us we need not require that he sample from the exact target distribution. Sampling from distributions which are ε -close to the desired one is just as good, for small but non-zero ε . This is also evidenced in the definitions of sampling classes as they are robust to small deviations from the target distribution. For this case of approximate sampling we can show the following:

Lemma 1. *For any sampling problem P which admits a GES, leaking at most the size of the index x specifying the distribution to be sampled from, it is the case that $P \in \text{SampMA}/\text{poly}$.*

The existence of a GES such that the client can obtain approximate samples from a distribution that is efficiently sampleable by a quantum computer reduces to asking whether $\text{SampBQP} \subset \text{SampMA}/\text{poly}$, for which we have:

Lemma 2. *If $\text{SampBQP} \subset \text{SampMA}/\text{poly}$ and Conjecture 1 is true then PH collapses at the 4th level.*

The complete proofs can be found in Appendix 7.4.

4 Quantum GES

Lastly, motivated by the existence of UBQC for BQP problems, we would like to know which functions admit a quantum GES or QGES. In a sense, this section is dedicated to ‘quantizing’ the Abadi et al. result. First of all, we need to define what a QGES is:

Definition 7 (Quantum Generalised Encryption Scheme (QGES)). *A quantum generalised encryption scheme (QGES) is a two party protocol between a quantum client C , and an unbounded server S , characterized by:*

- *A function $f : \Delta \rightarrow \Sigma$, where $\Delta, \Sigma \subseteq \{0, 1\}^*$.*
- *A cleartext input $x \in \text{Domain}(f)$, for which the client wants to compute $f(x)$.*
- *An expected polynomial-time key generation algorithm K which works as follows: for any $x \in \text{Domain}(f)$, with probability greater than $1/2 + 1/\text{poly}(|x|)$ we have $(k, \text{success}) \leftarrow K(x)$, where $k \in \{0, 1\}^{\text{poly}(|x|)}$. If the algorithm does not return success then we have $(k', \text{fail}) \leftarrow K(x)$, where $k' \in \{0, 1\}^{\text{poly}(|x|)}$.*
- *A quantum polynomial-time algorithm QE , that takes as input classical bit strings and produces as output a quantum state, which works as follows: for any $x \in \text{Domain}(f)$, $k \in \{0, 1\}^{\text{poly}(|x|)}$ we have that $|y\rangle \leftarrow QE(x, k)$, where $|y\rangle \in \mathcal{H}$ and $\dim(\mathcal{H}) = 2^{\text{poly}(|x|)}$.*
- *A polynomial-time deterministic algorithm E which works as follows: for any $x \in \text{Domain}(f)$, $k \in \{0, 1\}^{\text{poly}(|x|)}$ and $s \in \{0, 1\}^{\text{poly}(|x|)}$ we have that $w \leftarrow E(x, k, s)$, where $w \in \{0, 1\}^{\text{poly}(|x|)}$.*
- *A polynomial-time deterministic decryption algorithm D , which works as follows: for any $x \in \text{Domain}(f)$, $k \in \{0, 1\}^{\text{poly}(|x|)}$ and $s \in \{0, 1\}^{\text{poly}(|x|)}$ we have that $z \leftarrow D(s, k, x)$, where $z \in \{0, 1\}^{\text{poly}(|x|)}$.*

And satisfying the following properties:

1. *There are m rounds of communication, such that $m = \text{poly}(|x|)$. Denote the client’s message in round i as c_i and the server’s message as s_i .*
2. *On cleartext input x , C runs the key generation algorithm until success to compute a key $(k, \text{success}) = K(x)$. This happens before the communication between C and S is initiated, and the key k is used throughout the protocol. C then runs $QE(x, k)$ to obtain a quantum encryption of the input, $|y\rangle$ and sends it to S .*

3. In round i of the protocol, C computes $c_i = E(x, k, \bar{s}_{i-1})$, where \bar{s}_{i-1} denotes the server's responses up to and including round $i-1$, i.e. $\langle s_0, s_1 \dots s_{i-1} \rangle$. We assume that s_0 is the empty string. C then sends c_i to S .
4. In round i of the protocol, S responds with s_i , such that $s_i \in \{0, 1\}^{\text{poly}(|x|)}$. Additionally, the server's responses are drawn probabilistically from a distribution which is consistent with property 5.
5. At the end of the protocol, C computes $z \leftarrow D(\bar{s}_m, k, x)$ and with probability $1/2 + 1/\text{poly}(|x|)$, we have that $z = f(x)$.

The definition of QGES is similar to both that of the GES and that of UBQC. In fact, it is easy to see the following:

Lemma 3. *UBQC is a QGES for $f \in \text{BQP}$ leaking at most the size of the input x .*

The proof can be found in Section 7.5. UBQC is in fact an instance of a more particular type of QGES as it has the property of being an *offline* protocol. What this means is that the client can send a quantum state to the server, representing an encryption of the input, and decide afterwards which input it intends to use. Essentially the client is free to change its mind about the input and not commit to a particular input when the protocol commences. More formally, we define offliness as follows:

Definition 8 (Offline QGES). *Let $x_1, x_2 \in \text{Domain}(f)$ be two different inputs for f ($x_1 \neq x_2$) and let $|y\rangle \leftarrow QE(x_1, k)$ be a quantum encryption of x_1 with some compatible key k . A QGES is offline if there exists a polynomial-sized quantum circuit which the client can apply locally on her system after having sent $|y\rangle$ to the server, such that the state of her system and that of the servers are compatible with her having chosen as input x_2 .*

One might ask whether this property is not immediately satisfied by a QGES leaking only the size of the input. Indeed, in the classical case, any encrypted string sent by the client to the server must be compatible with all possible inputs of the same size. In other words, there exists an efficient update procedure that the client can perform in order to switch from one input to another. Thus, a GES leaking only the size of the input is implicitly offline in this view.

But the situation is different in the quantum case. The condition that the QGES leaks only the size of the input to the server is equivalent to saying that the density matrix corresponding to the quantum encryption, which the server receives, is the same for all inputs of the same size. Since the density matrix is the same, that means that there exists a unitary which can map from one purification of this state, corresponding to one input, to another, corresponding to a different input. This unitary must be verifiable in the sense that the client can check (using a quantum SWAP test) whether the unitary maps to the correct purification. In the classical case, this is sufficient to ensure that the procedure is efficient, since the mapping is just flipping the bits of one encryption key into another. In the quantum case, however, this unitary need not have a polynomial-sized quantum circuit representation.

Offliness simply imposes that such a circuit exist. UBQC trivially satisfies this property, since, no matter which input the client wants to use, it will always send random $|+\theta\rangle$ states to the server. In other words, the procedure $QE(x, k)$ does not depend explicitly on x , only on $|x|$. Because we know that functions which admit a GES are contained in the class $\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$, it is natural to ask what kind of containment we can find for functions which admit an offline QGES such as UBQC. This leads us to the following result which we prove in Appendix 7.6:

Theorem 4. *If f admits an offline QGES leaking at most the size of the input, then $f \in \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$.*

Of course, just like with Theorem 1, one can ask whether the class of interest should in fact be $\text{BQP}^{\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}}$ since the BQP client is using the QGES as an oracle. But just

as $\text{BPP}^{\text{NP}/\text{poly}} \cap \text{coNP}/\text{poly} = \text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$ it is the case that $\text{BQP}^{\text{QCMA}/\text{qpoly}} \cap \text{coQCMA}/\text{qpoly} = \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$, which we show at the end of Appendix 7.6. Note that the client need not be a fully BQP-capable machine (and indeed, for UBQC the only quantum capabilities of the client are to prepare single qubits), however this is the most general way in which we can model it in our QGES.

Another aspect worth addressing is what happens if we drop the “offline” requirement. As mentioned, that would imply that the mapping from one purification of the encrypted quantum state to another can in principle be any unitary operation so long as the client can check that the correct mapping was performed (with high probability). Having such a weak restriction on this unitary makes it very difficult to impose an upper bound on the types of computations that are allowed by such a scheme. Indeed, the offliness condition plays a crucial role in our proof of Theorem 4. At the same time, it is arguably a very natural condition to have in any realistic protocol. We therefore leave the online case as an open problem.

Theorem 4 can be viewed as a quantum version of Theorem 1 which, as mentioned, was used by Abadi et al. to show that there can be no GES for NP-hard functions unless the polynomial hierarchy collapses. As we have stated before, for quantum computers, the possibility of delegating NP-complete problems makes, arguably, even more sense since Grover’s algorithm offers a quadratic speed-up in solving such problems [33]. Alas, we show that even with a QGES, delegating such problems seems unlikely. This is because of the following result:

Theorem 5. *There can be no offline QGES, leaking at most the size of the input, for NP-hard functions unless $\Pi_3^P \subseteq \text{NP}^{\text{NP}^{\text{PromiseQMA}}}$.*

This is as close to a collapse of the polynomial hierarchy as one can reasonably hope to get, given a quantum hypothesis. The proof is provided in Appendix 7.7.

Note that in the definition of offline QGES we merely assumed that there exists some efficient quantum circuit which the client could apply to map one input to another. However, we never explicitly stated that the client could come up with this circuit in polynomial time. If we also added this condition then we would find that $f \in \text{BQP}/\text{qpoly}$ (which is of course contained in $\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$). In this case, using a result of Aaronson and Drucker which is a quantum version of the Karp-Lipton theorem [58], it follows that having such a QGES for NP-hard functions leads to $\Pi_2^P \subseteq \text{QMA}^{\text{PromiseQMA}}$. Our proof of Theorem 5 uses similar techniques and in fact strengthens the result of Aaronson and Drucker from $\text{NP} \subset \text{BQP}/\text{qpoly}$ implies $\Pi_2^P \subset \text{QMA}^{\text{PromiseQMA}}$, to $\text{NP} \subset \text{BQP}/\text{qpoly}$ implies $\Pi_2^P \subseteq \text{NP}^{\text{PromiseQMA}}$.

5 Implications for quantum verification

So far we have focused on protocols in which a client delegates a computation to a quantum server while maintaining some level of privacy about the input to the computation. Of course, in a realistic scenario the client would also be interested in the correctness of the computation. In other words, even if the server does not learn the client’s input he might deviate from the protocol and return incorrect results. How could the client detect such malicious behaviour?

When the client has a single-qubit preparation device, Fitzsimons and Kashefi showed that the UBQC protocol can be made verifiable [21]. The extension they made was to embed certain “trap qubits” into the graph state of the server. These traps are qubits which the server will measure in the same basis in which they were prepared. This means that their outcomes are deterministic and known only to the client. The key element is that the server cannot distinguish trap qubits from the rest of the states in the computation because of the blindness property provided by UBQC. Therefore the client can leverage this to test that the server performs his entanglement and measurement operations correctly. This ensures that with high probability the MBQC computation was successful. A schematic picture of this protocol, adapted from [17], is shown in Figure 2.

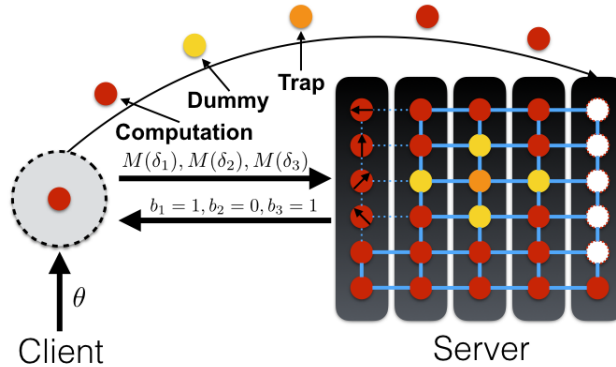


Figure 2: Fitzsimons-Kashefi verification protocol

A number of other protocols have been proposed for the verification of quantum computations, however they all either require some quantum communication between the client and the server, or that the server is able to perform PSPACE-complete computations, or that the client is interacting with more than one server [15, 32, 46, 47, 59]. The major open problem is whether a fully classical client can verify the computations performed by a single server which is restricted to BQP computations [60].

Because of the success of transforming UBQC into a verification protocol the hope was that if one could develop a classical client version of UBQC then that protocol could also be made verifiable. There was, in fact, another reason to favour such an approach, stemming from the difference between classical and quantum authentication. To give a brief description, an authentication protocol involves two parties a sender, S , and a receiver, R sharing a common secret key. S would like to send a message to R through an insecure channel such that R has the guarantee that the message came from S and was not modified or replaced by some malicious party. For the case of classical authentication, where the message, the key and the channel are all classical, there exist protocols in which the message is not encrypted. Surprisingly, however, for quantum authentication Barnum et al. showed that the message sent by S to R must be encrypted [61]. In other words, if the message were not encrypted then an eavesdropper would be able to convince the receiver to accept an incorrect message. We can see that in authentication we are in fact dealing with a verification task. The receiver must verify the authenticity of his received message.

In fact, the similarity between verification and authentication is even more noticeable in one of the first protocols for verifying quantum computations which relied on a modified authentication scheme [15]. This protocol, by Aharonov et al. is based on the observation that verification can be viewed as the client trying to authenticate the state $U|\psi\rangle$, where $|\psi\rangle$ is the client's input and U is the computation that the server should perform. This is illustrated in Figure 3. One could posit that the similarity between verification and authentication would imply that quantum verification requires some form of encryption. However, it has been shown that this is not true. A recent protocol by Morimae and Fitzsimons for post-hoc quantum verification is able to verify arbitrary BQP computations without the use of blindness or encryption [47]. Nonetheless, it is still interesting to investigate whether it is possible to have a blind scheme for classical client verification.

Our result suggests that the answer is no. We have shown that the existence of a classical client UBQC is contingent on the unlikely inclusion $\text{BQP} \subset \text{NP/poly} \cap \text{coNP/poly}$, in the case of decision problems, and the arguably more unlikely collapse of the polynomial hierarchy for sampling problems. Therefore, should a protocol exist for classical client, single server quantum verification, it will probably not rely on blindness as it does for UBQC. Some plausible alternatives are the following:

- As with the proof that $\text{IP} = \text{PSPACE}$ and hence that any PSPACE computation admits an interactive protocol (albeit one requiring the server to do PSPACE-complete computations), it might be possible to have a similar protocol in which the server is restricted to BQP

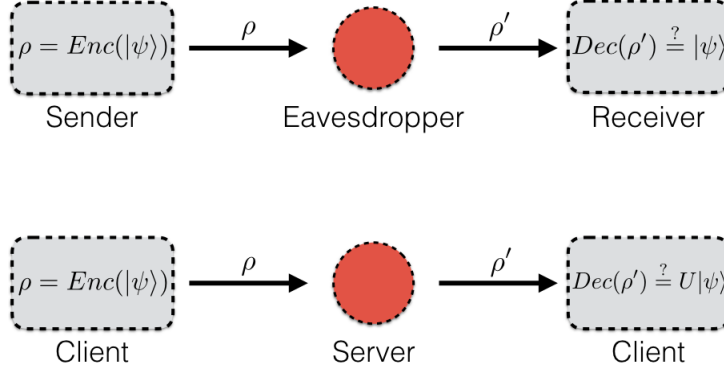


Figure 3: Similarity between quantum authentication and verification

computations [59]. The protocol would reveal the input and the computation to the server, however the client would rely on the specific structure of the computation in order to check certain properties that would determine its correctness.

- A recent protocol by Mantri et al. demonstrates how a completely classical client can delegate an MBQC computation to a quantum server while retaining some secrecy about which computation is being performed [44]. The protocol relies on the fact that measurements in an MBQC graph are only partially ordered and there are multiple total orderings compatible with this. This so-called flow ambiguity means that the server is uncertain about which computation is being performed. Unlike UBQC, however, the client leaks more information to the server than just the size of the input and computation.
- Both UBQC and the generalised encryption schemes considered in our paper assume information-theoretic security. It is not implausible to imagine that one could design a verification protocol which is “merely” computationally secure. Indeed, recent protocols for quantum fully homomorphic encryption do rely on assumptions about the hardness of certain lattice problems even in the face of quantum computers [28, 38, 40]. Additionally, for a more restricted class of quantum computations called instantaneous quantum computations, IQP, Shepherd and Bremner proposed a procedure whereby a classical client can determine whether the server has sampled from an IQP distribution. Their protocol also relies on computational assumptions such as the hardness of a problem involving binary matroids [45].

6 Conclusions

The recent surge in development for both cryptography and quantum computation has seen significant overlap between the two fields. As scalable quantum computers get closer to reality we find ourselves having to design more clever protocols for performing cryptographic tasks in the presence of quantum computers. It therefore came as a surprise when Universal Blind Quantum Computing, introduced almost a decade ago in 2009, showed that quantum computation on encrypted data is possible with information-theoretic security and minimal quantum requirements for the client.

Similar to the classical setting, quantum computing on encrypted data, whether through UBQC or other techniques, has led to several novel directions such as verification of outsourced quantum computations [21, 27] as well as quantum fully homomorphic encryption [28, 38]. This represents a key application for emerging quantum technologies as it provides information-theoretic security for delegated computation, whereas classical approaches seem to fail in doing so. Of course, the trade off is that in all of these cases the client is required to prepare or measure single qubit states. Is it really the case that no classical encryption scheme leads to the same functionality? Does it mean that information-theoretically secure verification of encrypted computation or information-theoretically secure QFHE could not be obtained through a classical encoding?

The central result of our paper was to show that this is indeed the case, unless some very unlikely containments of complexity classes occur. We relied heavily on the major results of Abadi et. al that bridged computational complexity and efficient hiding by showing that no NP-hard function could be delegated efficiently and securely by a BPP client, unless the polynomial hierarchy collapses. Similarly, our result links the concept of quantum hiding with the complexity of computing and provides a new approach towards the characterisation of quantum complexity classes within the same formalism.

We also provided an upper bound for the types of functions computable through UBQC and the more general QGES. We saw that the addition of quantum communication makes the QGES more powerful than its classical counterpart allowing for the delegation of BQP computations. However, it seems no more powerful than a GES at delegating NP-hard functions. The latter result is essentially a quantized version of the Abadi et. al no-go theorem. Additionally, we explained how this impacts the design of any future protocol for classical client verification of quantum computation in the single server case. Such a protocol would have to either not be blind in the UBQC sense, or not enforce information-theoretic security.

As open problems we mention the following:

- Is it possible to strengthen our result from Theorem 2 to provide an oracle separation between BQP and NP/poly?
- We showed that functions which admit an offline QGES are contained in QCMA/qpoly \cap coQCMA/qpoly. What upper bound can be placed on functions which admit an online QGES?
- What if we consider a QGES in which the client’s quantum message is logarithmic in the size of the input (while the classical communication is still polynomial)? Can such a scheme allow for the evaluation of arbitrary BQP functions?
- Suppose we relax the demand for information-theoretic security, and settle for security under a cryptographic assumption. In that case, is it possible to do blind universal QC using only classical communication between client and server?

Acknowledgements

We would like to thank the following people for useful discussions and comments: Petros Wallden, Matty J Hoban, Kousha Etessami, Marc Kaplan, Ronald de Wolf, Urmila Mahadev, Umesh Vazirani and Pia Kulik. We are especially grateful to Urmila Mahadev and Umesh Vazirani for pointing out a mistake in the initial proof of Theorem 4, which we have corrected. A.G. is in particular grateful to Petros Wallden and Matty J Hoban for their patience and explanations when answering several questions. E.K. acknowledges funding through EPSRC grant EP/N003829/1. S.A. is supported by a Vannevar Bush Fellowship from the US Department of Defense.

7 Appendix

7.1 Preliminaries

7.1.1 Quantum information and computation basics

In this subsection we provide a few basic notions regarding quantum information and quantum computation and refer the reader to the appropriate references for a more in depth presentation [62, 63].

A quantum state (or a quantum register) is a unit vector in a complex Hilbert space, \mathcal{H} . We denote quantum states, using standard Dirac notation, as $|\psi\rangle \in \mathcal{H}$, called a ‘ket’ state. The dual of this state is denoted $\langle\psi|$, called a ‘bra’, and is a member of the dual space \mathcal{H}^\perp . We will only

be concerned with finite-dimensional Hilbert spaces. Qubits are states in two-dimensional Hilbert spaces. Traditionally, one fixes an orthonormal basis for such a space, called *computational basis*, and denotes the basis vectors as $|0\rangle$ and $|1\rangle$. Gluing together systems to express the states of multiple qubits is achieved through *tensor product*, denoted \otimes . The notation $|\psi\rangle^{\otimes n}$ denotes a state comprising of n copies of $|\psi\rangle$. If a state $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ cannot be expressed as $|a\rangle \otimes |b\rangle$, for any $|a\rangle \in \mathcal{H}_1$ and any $|b\rangle \in \mathcal{H}_2$, we say that the state is *entangled*.

Quantum mechanics dictates that there are two ways to change a quantum state: *unitary evolution* and *measurement*. Unitary evolution involves acting with some unitary operation U (so $UU^\dagger = U^\dagger U = I$, where the \dagger operation denotes the hermitian adjoint, obtained through transposing and complex conjugating) on $|\psi\rangle$, thus producing the mapping $|\psi\rangle \rightarrow U|\psi\rangle$.

Measurement, in its most basic form, involves expressing a state $|\psi\rangle$ in a particular orthonormal basis, \mathcal{B} , and then choosing one of the basis vectors as the state of the system post-measurement. The index of that vector is the classical outcome of the measurement. The post-measurement vector is chosen at random and the probability of obtaining a vector $|v\rangle \in \mathcal{B}$ is given by $|\langle v|\psi\rangle|^2$. There are more general types of measurement, however this is the only type that is relevant to our paper.

States denoted by kets are also referred to as *pure states* as they are states of maximal information for a quantum system. In other words, having a pure state for a particular quantum system means knowing all there is to know about the state of that system. When maximal information is not available, states are referred to as *mixed* and can be represented using *density matrices*. These are positive semidefinite, trace one, hermitian operators. The density matrix of a pure state $|\psi\rangle$ is $\rho = |\psi\rangle\langle\psi|$.

An essential operation concerning density matrices is the *partial trace*. This provides a way of obtaining the density matrix of a subsystem that is part of a larger system. Partial trace is linear, and is defined as follows. Given two density matrices ρ_1 and ρ_2 with Hilbert spaces \mathcal{H}_1 and \mathcal{H}_2 , we have that:

$$\rho_1 = \text{Tr}_2(\rho_1 \otimes \rho_2) \quad \rho_2 = \text{Tr}_1(\rho_1 \otimes \rho_2) \quad (1)$$

In the first case one is ‘tracing out’ system 2, whereas in the second case we trace out system 1. This property together with linearity completely defines the partial trace. For if we take any general density matrix, ρ , on $\mathcal{H}_1 \otimes \mathcal{H}_2$, expressed as:

$$\rho = \sum_{i,i',j,j'} a_{ii'jj'} |i\rangle_1 \langle i'|_1 \otimes |j\rangle_2 \langle j'|_2 \quad (2)$$

where $\{|i\rangle\}$ ($\{|i'\rangle\}$) and $\{|j\rangle\}$ ($\{|j'\rangle\}$) are orthonormal bases for \mathcal{H}_1 and \mathcal{H}_2 , if we would like to trace out subsystem 2, for example, we would then have:

$$\text{Tr}_2(\rho) = \text{Tr}_2 \left(\sum_{i,i',j,j'} a_{ii'jj'} |i\rangle_1 \langle i'|_1 \otimes |j\rangle_2 \langle j'|_2 \right) = \sum_{i,i',j} a_{ii'jj} |i\rangle_1 \langle i'|_1 \quad (3)$$

An important result, concerning the relationship between mixed states and pure states which we use in our paper, is the fact that any mixed state can be purified. In other words, for any mixed state ρ over some Hilbert space \mathcal{H}_1 one can always find a pure state $|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ such that $\dim(\mathcal{H}_1) = \dim(\mathcal{H}_2)$ ⁵ and:

$$\text{Tr}_2(|\psi\rangle\langle\psi|) = \rho \quad (4)$$

Moreover, the purification $|\psi\rangle$ is not unique and so another important result is the fact that if $|\phi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2$ is another purification of ρ then there exists a unitary U , acting only on \mathcal{H}_2 (the additional system that was added to purify ρ) such that:

$$|\phi\rangle = (I \otimes U) |\psi\rangle \quad (5)$$

⁵One could allow for purifications in larger systems, but in our paper we restrict attention to same dimensions.

We will refer to this as the *purification principle*.

Quantum computation is most easily expressed in the *quantum gates model*. In this framework, gates are unitary operations which act on groups of qubits. As with classical computation, universal quantum computation is achieved by considering a fixed set of quantum gates which can approximate any unitary operation up to a chosen precision. The most common universal set of gates is given by:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} \quad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In order, the operations are known as Pauli X and Pauli Z, Hadamard, the T-gate and controlled-NOT. Note that general controlled- U operations are operations performing the mapping $|0\rangle|\psi\rangle \rightarrow |0\rangle|\psi\rangle$, $|1\rangle|\psi\rangle \rightarrow |1\rangle U|\psi\rangle$. The matrices express the action of each operator on the computational basis. A classical outcome for a particular quantum computation can be obtained by measuring the quantum state resulting from the application of a sequence of quantum gates.

The final notion which needs mentioning is the *quantum SWAP test*. This is a simple procedure for determining whether two quantum states $|\psi\rangle, |\phi\rangle \in \mathcal{H}$ are close to each other or far apart. We express closeness in terms of the absolute value of their inner product $|\langle\psi|\phi\rangle|$. The test involves preparing a qubit in the state $(|0\rangle + |1\rangle)/\sqrt{2}$ and performing a controlled-SWAP operation between that qubit and the state $|\psi\rangle|\phi\rangle$. SWAP is defined by the mapping $|\psi\rangle|\phi\rangle \rightarrow |\phi\rangle|\psi\rangle$, so we obtain the state:

$$\frac{|0\rangle|\psi\rangle|\phi\rangle + |1\rangle|\phi\rangle|\psi\rangle}{\sqrt{2}}$$

If one then applies a Hadamard operation to the first qubit and measures it in the computational basis it can be shown that the probability of obtaining outcome $|0\rangle$ is $(1 + |\langle\psi|\phi\rangle|^2)/2$.

7.1.2 Measurement-based quantum computation

As UBQC is expressed in the model of *measurement-based quantum computation* (MBQC), defined in [50, 51], we provide a brief description of this subject. Unlike the quantum gates model of computation, in MBQC a given computation is performed by measuring qubits from a large entangled state. Traditionally, this state consists of $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ qubits entangled using the CZ operation, where:

$$\text{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

They are then measured in the basis $(|+\phi\rangle, |-\phi\rangle)$, where $|\pm\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \pm e^{i\phi}|1\rangle)$. These measurements are denoted as $M(\phi)$, and depending on the value of ϕ chosen for each qubit one can perform universal quantum computation. For this to work, the entangled qubits need to form a *universal graph state*. Such a state consists, for example, of N qubits in the state $|+\rangle$. These qubits have been entangled according to some graph structure G , such that there exist measurement patterns (an order for measuring the qubits in G and the corresponding measurement angles) for each quantum computation consisting of $O(N)$ gates.

In other words, a universal graph state allows one to perform any quantum computation up to a certain size. An example of such a state is the *brickwork state*, defined in [14] from which we illustrate Figure 4. To be more precise, suppose we would like to perform some quantum computation described by a circuit consisting of N gates. The corresponding MBQC computation consists of the following steps:

1. **Initialization.** Prepare $O(N)$ qubits, each in the state $|+\rangle$.

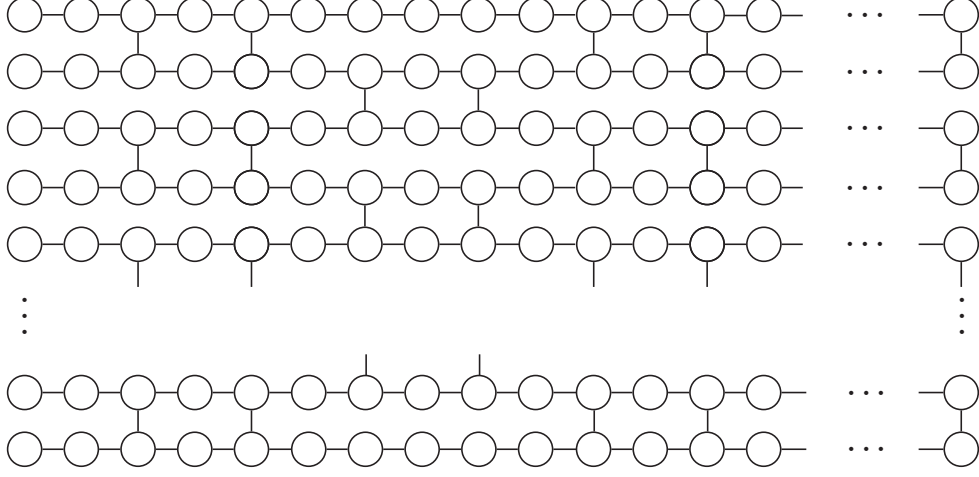


Figure 4: Brickwork state

2. **Entanglement.** Entangle the qubits according to some universal graph state structure, such as the brickwork state.
3. **Measurement.** Measure each qubit, i using $M(\phi_i)$, for some angle ϕ_i determined based on the computation we would like to perform. The angles ϕ_i are referred to as the *computation angles*.
4. **Correction.** Apply appropriate corrections (Pauli X and Z operations) to the qubits, based on the measurement outcomes.

The last two steps can be performed together. This is because if we would like to apply a Pauli X correction to a qubit, i , before measuring it, we can simply measure it using $M(-\phi_i)$. Similarly, if we would like to apply a Pauli Z correction to that same qubit we measure it using $M(\phi_i + \pi)$. Therefore, the general measurement performed on a particular qubit will be $M((-1)^s \phi_i + r\pi)$, where $s, r \in \{0, 1\}$ are determined by previous measurement outcomes.

As a final remark, it should be noted that one can translate quantum circuits into MBQC patterns in a canonical way. For instance, the universal gate set mentioned in the previous subsection, and hence any quantum circuit comprising of those gates, can be translated directly into MBQC. See for example [14] for more details.

7.1.3 Complexity theory

We refer the reader to the Complexity Zoo [30] for the definitions of standard complexity classes.

Throughout most of the paper we are working with advice classes such as P/poly or NP/poly . These two classes correspond to computations that can be performed by polynomial-time deterministic and non-deterministic Turing machines, receiving an advice string for each input length. More formally, for some complexity class \mathcal{C} , we say that a language L is in \mathcal{C}/poly if there exists a language $L' \in \mathcal{C}$ and a set of advice strings $\{a_n\}_{n \geq 1}$ with $|a_n| = \text{poly}(n)$ and $x \in L$ iff $(x, a_{|x|}) \in L'$. In other words, the machines will make use of an advice string of length polynomial in the size of the input that only depends on the size of the input. If we fix this polynomial to be $O(n^d)$, for some constant d , then the corresponding classes will be $P/O(n^d)$ or $NP/O(n^d)$.

We will also encounter MA/rpoly , where the advice is a polynomial-sized string that is drawn randomly from a probability distribution which depends only on the input size. So in the above definition, instead of having a set of advice string $\{a_n\}_{n \geq 1}$, we will have a set of probability distributions $\{\mathcal{P}_n\}_{n \geq 1}$, such that \mathcal{P}_n is a distribution over $\{0, 1\}^{\text{poly}(n)}$ and the advice received for some input x is drawn from $\mathcal{P}_{|x|}$. The decisions for acceptance or rejection of an input is now probabilistic, hence MA replaces NP . An important result concerning these classes, which we use in the paper, is the following:

Theorem 6 (Aaronson [64]). $\text{MA}/\text{rpoly} = \text{MA}/\text{poly} = \text{NP}/\text{poly}$

Lastly, concerning advice, we can also have quantum advice such as with the class QCMA/qpoly . In this case, we have a set of quantum states $\{\rho_n\}_{n \geq 1}$, such that ρ_n is a density matrix over a $2^{\text{poly}(n)}$ -dimensional Hilbert space (i.e. a state of polynomially many qubits), and for some input x the advice state will be $\rho_{|x|}$.

Complements of classes are denoted by adding the prefix **co**, so for instance coNP/poly is the complement of NP/poly .

We also refer to oracle classes in the paper. Briefly, an oracle is some black box which can be invoked by a Turing machine in order to obtain the solution to some problem in one time step. For example, the class of problems which can be solved by a deterministic polynomial-time Turing machine with access to some oracle $O : \{0, 1\}^* \rightarrow \{0, 1\}$ is denoted P^O . If O is an oracle for some NP -complete problem, then the corresponding class is P^{NP} . Oracles are used, among other things, for defining the polynomial hierarchy PH . So the zeroth level of the polynomial hierarchy is defined as $\Sigma_0^{\text{P}} = \Pi_0^{\text{P}} = \text{P}$, whereas an arbitrary level, k is defined by $\Sigma_k^{\text{P}} = \text{NP}^{\Sigma_{k-1}^{\text{P}}}$ and $\Pi_k^{\text{P}} = \text{coNP}^{\Sigma_{k-1}^{\text{P}}}$.

We now transition from decision problems to sampling problems. These are problems for which we specify a distribution and would like to obtain a sample from that distribution. Formally, we state the definition from [65] of polynomial-time classical and quantum sampling:

Definition 9 ([65] Sampling Problems, **SampP**, and **SampBQP**). *A sampling problem S is a collection of probability distributions $(\mathcal{D}_x)_{x \in \{0, 1\}^*}$, one for each input string $x \in \{0, 1\}^n$, where \mathcal{D}_x is a distribution over $\{0, 1\}^{p(n)}$, for some fixed polynomial p . Then **SampP** is the class of sampling problems $S = (\mathcal{D}_x)_{x \in \{0, 1\}^*}$ for which there exists a probabilistic polynomial-time algorithm B that, given $\langle x, 0^{1/\varepsilon} \rangle$ as input, samples from a probability distribution \mathcal{C}_x such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$. **SampBQP** is defined the same way, except that B is a quantum algorithm rather than a classical one.*

Similar definitions can be given for **SampMA**, **SampQMA** and so on. See for example Definition 11. An important theorem regarding the possibility of having efficient classical procedures for sampling distributions resulting from quantum circuits was proven by Aaronson and Arkhipov:

Theorem 7 ([31] Aaronson, Arkhipov). ***SampP** = **SampBQP** implies that estimating the permanent of a Gaussian matrix is in BPP^{NP} .*

The latter statement is believed to be unlikely since it is known that computing the permanent of a matrix is a $\#\text{P}$ -complete problem and it is conjectured that estimating the permanent is equally as hard:

Conjecture 1 ([31, 65] Aaronson, Arkhipov). *Estimating the permanent of a Gaussian matrix is $\#\text{P}$ -complete.*

The proof of Theorem 7 relies on the BosonSampling problem. In BosonSampling, identical photons (bosons) are sent through a linear optics network and non-adaptive measurements are performed to count the number of photons in each mode. The network can be specified by a matrix, denoted A , from a set of $m \times n$ column-orthonormal matrices $\mathcal{U}_{m,n}$. Here m denotes the number of modes in the circuit and n the number of photons. One then also specifies a basis state $S = (s_1, \dots, s_m)$, where s_i denotes the number of photons in mode i (so $s_1 + \dots + s_m = n$). This then defines the distribution \mathcal{D}_A representing the probability of measuring the state S in the circuit defined by A (i.e. measuring s_i photons in mode i for all modes). BosonSampling is therefore the problem of sampling either exactly or approximately from this distribution [31]. Aaronson and Arkhipov then define a BosonSampling oracle which we also use in our results and so we include the definition here:

Definition 10 ([31] **BOSONSAMPLING** oracle). *Let \mathcal{O} be an oracle that takes as input a string $r \in \{0, 1\}^{\text{poly}(n)}$, an $m \times n$ matrix $A \in \mathcal{U}_{m,n}$, and an error bound $\varepsilon > 0$ encoded as $0^{1/\varepsilon}$. Also,*

let $\mathcal{D}_{\mathcal{O}}(A, \varepsilon)$ be the distribution over outputs of \mathcal{O} if A and ε are fixed but r is uniformly random. We call \mathcal{O} an exact **BOSONSAMPLING** oracle if $\mathcal{D}_{\mathcal{O}}(A, \varepsilon) = \mathcal{D}_A$ for all $A \in \mathcal{U}_{m,n}$. Also, we call \mathcal{O} an approximate **BOSONSAMPLING** oracle if $\|\mathcal{D}_{\mathcal{O}}(A, \varepsilon) - \mathcal{D}_A\| \leq \varepsilon$ for all $A \in \mathcal{U}_{m,n}$ and $\varepsilon > 0$.

The oracle is a deterministic function which takes as input a description of the optical network as well as the randomness used for sampling. Importantly \mathcal{O} uses no intrinsic randomness other than the one provided as input. The final result which we utilize from [31] is:

Theorem 8 ([31] Aaronson, Arkhipov). *If \mathcal{O} is the oracle from Definition 10 then $\mathbf{P}^{\#P} \subseteq \mathbf{BPP}^{\mathbf{NP}^{\mathcal{O}}}$.*

In other words, if a **BPP** machine can simulate the behavior of \mathcal{O} then $\mathbf{P}^{\#P} \subseteq \mathbf{BPP}^{\mathbf{NP}}$, but since $\mathbf{PH} \subseteq \mathbf{P}^{\#P}$, from Toda's theorem [66], it would follow that the polynomial hierarchy is contained in $\mathbf{BPP}^{\mathbf{NP}}$ which implies a collapse at the third level (because $\mathbf{BPP} \subseteq \Sigma_2^P$ by the Sipser-Gács-Lautemann theorem [67]).

7.2 Proof of Theorem 1

This proof is a simplified version of the one from [29].

Proof. Suppose that f admits a GES which leaks at most the size of the input. To show that $f \in \mathbf{NP}/\text{poly} \cap \mathbf{coNP}/\text{poly}$ we prove that $f \in \mathbf{NP}/\text{poly}$ by constructing an \mathbf{NP}/poly algorithm for f which can also compute the complement of f (thus also proving containment in $\mathbf{coNP}/\text{poly}$). We start by first considering the one round case. In other words, the protocol works as follows:

1. The client runs $K(x)$ until success to produce an encryption key k .
2. The client computes the encrypted string $y \leftarrow E(x, k, '')$ (where the last entry is the empty string) and sends it to the server.
3. The server sends a response r .
4. The client decrypts his response obtaining $z \leftarrow D(r, k, x)$. With probability greater than $1/2 + 1/\text{poly}(|x|)$ we have that $z = f(x)$.

Given that this is true, consider the following algorithm which takes x as input and produces $f(x)$ with probability greater than $1/2 + 1/\text{poly}(|x|)$:

- Denoting $|x| = n$, the algorithm receives as advice some string $x_n \in \text{Domain}(f)$ such that $|x_n| = n$ as well as r_n , where r_n is the server's response when being sent $y_n \leftarrow E(x_n, k_n, '')$. Here k_n is simply some key which can be used to encrypt x_n . The only reason we include x_n as part of the advice is so that we can check if $x_n = x$. If this is the case then the algorithm simply decrypts r_n obtaining $f(x)$ with high probability. The next steps assume that $x_n \neq x$.
- From the assumption that the GES leaks at most the size of the input, there must be some key k , such that $y_n \leftarrow E(x, k, '')$. This is because if there did not exist such a key, then if the server received y_n he would know that the input could not be x and hence more information would be leaked. Since $|k| = \text{poly}(n)$, the algorithm can non-deterministically search for k .
- The algorithm now simply computes $z \leftarrow D(r_n, k, x)$, which by definition of the GES, will be $f(x)$ with probability greater than $1/2 + 1/\text{poly}(n)$.

We have therefore given an \mathbf{MA}/rpoly algorithm for computing $f(x)$. The \mathbf{MA} part comes from the non-deterministic search for k and the fact that the algorithm is probabilistic. The advice is rpoly because the server's response is drawn from some probability distribution (which depends only on the length of the input). However, we know from Theorem 6 that $\mathbf{MA}/\text{rpoly} = \mathbf{NP}/\text{poly}$,

therefore $f \in \text{NP/poly}$. Since we can do the same thing for the complement, we also have that $f \in \text{coNP/poly}$ ⁶.

We now need to generalize this to the case where the client and the server interact for a polynomial number of rounds. Because the protocol is leaking at most the size of the input, denoted n , any transcript of the protocol will only depend on n . Therefore we can make the algorithm's advice to be a complete transcript of the protocol drawn from the distribution of all possible transcripts for input of length n . We would then again search non-deterministically for a key k which would make the input x compatible with this transcript. From the definition of the GES this again guarantees that we obtain the right outcome with probability $1/2 + 1/\text{poly}(|x|)$.

In the original paper of Abadi et al., the proof of this step is more elaborate and uses universal hash functions [29]. The reason is that, at the time, the result $\text{MA/rpoly} = \text{NP/poly}$ was unknown and so their proof uses hash functions in order to “derandomize” the algorithm and produce the correct value for $f(x)$ deterministically. \square

As mentioned, since the client is a BPP machine and the GES is capable of solving $\text{NP/poly} \cap \text{coNP/poly}$ problems, we would like to show that:

Lemma 4. $\text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}} = \text{NP/poly} \cap \text{coNP/poly}$

Proof. Clearly $\text{NP/poly} \cap \text{coNP/poly} \subseteq \text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}}$, so what we need to show is that $\text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}} \subseteq \text{NP/poly} \cap \text{coNP/poly}$. To do this, we first use Adleman's theorem [49], that $\text{BPP} \subseteq \text{P/poly}$, which we know is relativizing and have that $\text{BPP}^{\text{NP/poly} \cap \text{coNP/poly}} \subseteq \text{P/poly}^{\text{NP/poly} \cap \text{coNP/poly}}$. Next, it is easy to show that $\text{P/poly}^{\text{NP/poly} \cap \text{coNP/poly}} \subseteq \text{P}^{\text{NP/poly} \cap \text{coNP/poly}}$. This is because the advice received by the P/poly machine can just as easily be obtained from the $\text{NP/poly} \cap \text{coNP/poly}$ oracle. In other words, for any given input x and advice a for the P/poly machine, the P machine can simply query the $\text{NP/poly} \cap \text{coNP/poly}$ oracle with x in order to obtain the same advice a ⁷. It then simulates the P/poly machine.

We have therefore reduced our problem to showing that $\text{P}^{\text{NP/poly} \cap \text{coNP/poly}} \subseteq \text{NP/poly} \cap \text{coNP/poly}$. This can be done by adapting Brassard's proof [48] that $\text{P}^{\text{NP} \cap \text{coNP}} = \text{NP} \cap \text{coNP}$. The essential part of that proof is to show that $\text{P}^{\text{NP} \cap \text{coNP}} \subseteq \text{NP}$, while the containment in coNP follows by complementation. The idea is that for any $\text{P}^{\text{NP} \cap \text{coNP}}$ algorithm, A , deciding some language, we can devise an NP algorithm, NA , which also decides that language.

The NA algorithm will simulate A until it makes a query to the $\text{NP} \cap \text{coNP}$ oracle. At this point NA can non-deterministically guess the response to this query. To do so, note that if some language $L \in \text{NP} \cap \text{coNP}$ then it is the case that $L \in \text{NP}$ and $L^c \in \text{NP}$, where L^c is the complement of L . In other words, there exist non-deterministic algorithms N_L and N_{L^c} for deciding L and L^c , respectively. Assuming A 's query is for the language L , NA will simulate N_L , and for each non-deterministic branch of this simulation it will then also simulate N_{L^c} . Since L and L^c are complementary, it cannot happen that both the N_L and the N_{L^c} parts of the branches are accepting. We will therefore have branches in which both N_L and N_{L^c} were rejecting and branches in which either N_L was accepting or N_{L^c} was accepting. These latter branches determine the answer to the query for the $\text{NP} \cap \text{coNP}$ oracle. The NA algorithm will continue simulating A on these branches and reject on all others.

We can see that the above reasoning would also work if the oracle was $\text{NP/poly} \cap \text{coNP/poly}$ and the algorithm NA were an NP/poly algorithm receiving some advice string whose length is polynomial in the size of the input. Our modified NA can continue to simulate the oracle queries if we assume that the advice it receives is the concatenation of advices received by the $\text{NP/poly} \cap \text{coNP/poly}$ oracle for all queries. Since the number of queries is polynomial, the concatenation will also be polynomially bounded and hence constitutes a valid advice string for

⁶As mentioned in the main text, these containments are valid whenever f outputs one bit, corresponding to a decision problem. For general functions having a larger range, the technically correct containments would be in the relational versions of NP and coNP .

⁷The fact that the oracle responds with a single bit (acceptance or rejection) is not a problem, since the P machine can query the oracle for each bit of a .

an NP/poly algorithm. Therefore $\text{P}^{\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}} \subseteq \text{NP}/\text{poly}$ and through complementation $\text{P}^{\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}} \subseteq \text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$.

Because $\text{BPP}^{\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}} \subseteq \text{P}^{\text{NP}/\text{poly} \cap \text{coNP}/\text{poly}}$, our result follows immediately. \square

7.3 Proof of Theorem 2

In order to prove Theorem 2 we will construct an oracle using a version of the complement of Simon's problem [11]. Simon's problem is the following: given a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ (for some $n \in \mathbb{N}$) which is promised to be either 1-to-1 or have Simon's property (f is 2-to-1 and there exists some $s \in \{0, 1\}^n$, $s \neq 0^n$, such that for $x \neq y$, $f(x) = f(y)$ iff $x = s \oplus y$), decide which is the case. In particular, for Simon's problem, the deciding algorithm should accept if the function has Simon's property and reject if is a 1-to-1 function. The complement of this problem simply flips these two conditions. If one is not given an explicit description of f but restricts access to this function through an oracle then Simon's problem can be used to separate BPP from BQP . To be precise, the oracle is some function $O : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that for $n \in \mathbb{N}$, if we consider O restricted to the domain $\{0, 1\}^n$, denoted $O_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$, O_n is either a 1-to-1 function or a function satisfying Simon's property. A language which is then contained in BQP^O but not in BPP^O is $L(O) = \{0^n | O_n \text{ is a function with Simon's property}\}$ as shown in [11]. In fact, the complement of this language⁸ can be used to separate BQP^O and NP^O [54]. The reason for this is that while there is a witness for a function with Simon's property, namely the string s , there does not exist a witness for a general 1-to-1 function (in the relativized setting).

In our case, we would like to separate NP/poly^O and BQP^O . The intuition is the following: instead of considering a function O_n for each input length n , we consider a function O_x , for each input string $x \in \{0, 1\}^n$. In other words, for a fixed input length, n , there will be 2^n functions which need to be decided. But the NP^O machine receives only a polynomial amount of advice, which is the same for all of these 2^n functions. Therefore this advice should be insufficient to help the NP^O machine in deciding all of these inputs. Formalizing this intuition for any polynomial is problematic, as will become clear later in this appendix. Instead, if we fix the degree of the polynomial, then we can prove a separation.

We start by showing the following:

Lemma 5. *There exists an oracle O , based on the complement of Simon's problem, such that $\text{BQP}^O \not\subseteq \text{NP}^O$.*

Proof. The separation of BQP and NP with respect to an oracle has been shown a number of times before, [10, 68, 69], including with the complement of Simon's problem. However, we prove this lemma for our particular version of Simon's problem where instead of assigning a function to each input length, we assign different functions to different inputs.

We proceed by defining an oracle O and a language which we refer to as the complement of Simon's problem or $\text{coSimon}(O)$, such that $\text{coSimon}(O) \in \text{BQP}^O$ and $\text{coSimon}(O) \notin \text{NP}^O$. We start with the latter as it also clarifies what the oracle should do:

$$\text{coSimon}(O) = \{\langle 1^n, i \rangle | i \in \{0, 1\}^n \text{ and } f \text{ defined as } f(x) \equiv O(1^n, i, x) \text{ is a 1-to-1 function}\} \quad (6)$$

Strictly speaking, the problem we are defining is a promise problem, so the set defined above is the set of *yes* instances to the problem, whereas the set of *no* instances is not the complement but the set:

$$\{\langle 1^n, i \rangle | i \in \{0, 1\}^n \text{ and } f \text{ defined as } f(x) \equiv O(1^n, i, x) \text{ is a Simon function}\} \quad (7)$$

Here, by "*Simon function*" we mean a function having Simon's property.

It is clear from this definition that the oracle O is the one providing the functions for which we want test whether they are 1-to-1 or have Simon's property. Of course, the whole point is to

⁸Note that Simon's problem is a promise problem, so when speaking about the complement of $L(O)$ we are in fact referring to $L^c(O) = \{0^n | O_n \text{ is a 1-to-1 function}\}$.

restrict access to the descriptions of those functions and force the algorithm solving the problem to perform queries to the oracle. It is also clear that for any such O , $coSimon(O)$ will be contained in BQP^O since we can just run Simon's algorithm on the given input and flip acceptance and rejection. The algorithm will be linear in the size of the input. As is standard in quantum query complexity, we assume that the behaviour of the quantum oracle is to perform the unitary operation $|1^n\rangle |i\rangle |x\rangle |y\rangle \xrightarrow{O} |1^n\rangle |i\rangle |x\rangle |O(1^n, i, x) \oplus y\rangle$.

In general, the oracle O can be viewed as some function taking as input the tuple (n, i, x) and outputting $f_i(x)$, where $f_i : \{0, 1\}^n \rightarrow \{0, 1\}^n$ is a function which is either bijective or has Simon's property. Essentially n , which is given in unary, specifies the domain size of our functions, i is an index for a particular function and x is the value on which we evaluate f_i . These last two elements of the tuple are specified in binary and the oracle should be defined for all $n \in \mathbb{N}$ and all $i, x \in \{0, 1\}^n$. We will denote the set of functions used by the oracle for domain size n as \mathcal{F}_n , in other words:

$$\mathcal{F}_n = \{f_i | i \in \{0, 1\}^n \text{ and } f_i \text{ is defined as } f_i(x) \equiv O(1^n, i, x)\} \quad (8)$$

What are we going to do next is construct an *adversarial oracle* O , which is equivalent to defining the family of sets $\{\mathcal{F}_n\}_{n \in \mathbb{N}}$, in such a way that every non-deterministic Turing machine using the oracle O fails to decide correctly $coSimon(O)$. This is a standard diagonalization argument.

Since the set of non-deterministic Turing machines is countable we consider the k 'th machine, M_k , and check its behaviour when $n = k + n_0$, for some $n_0 \geq 0$ which we define later on. Suppose we take some index $i \in \{0, 1\}^n$, and tentatively make the i 'th function in \mathcal{F}_k a 1-to-1 function. By simulating the behaviour of M_k on this input we can check to see whether it accepts or rejects. If it rejects, then we are done, since M_k will incorrectly decide this input. Conversely, if M_k accepts, then by definition there exists a polynomial-sized path, in the non-deterministic computation tree of the machine, which leads to acceptance. We denote this path as π , and denote the length of π as $l = poly(n)$. M_k can make at most l queries to O on this path which we can represent as a list of tuples: $[(x_1, f_i(x_1)), (x_2, f_i(x_2)) \dots (x_l, f_i(x_l))]$, where x_1, \dots, x_l are the queried variables. An example of such a path is shown in Figure 5.

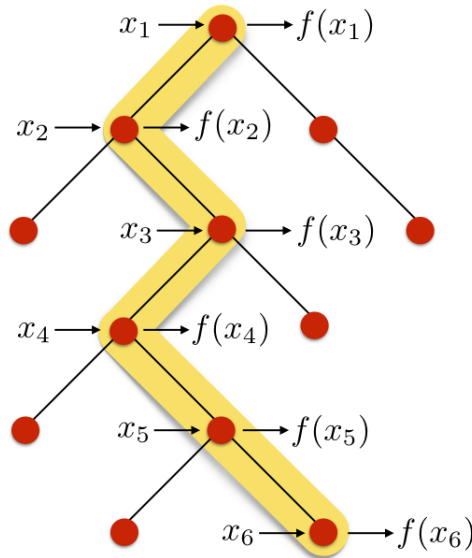


Figure 5: Computation tree with queries

We now simply consider a Simon function f' such that $f'(x_1) = f_i(x_1), \dots, f'(x_l) = f_i(x_l)$. How do we know such a function exists? The number of possible bit masks s such that $f(x) = f(x \oplus s)$ is $2^n - 1$ (since 0^n is excluded). By having f' match f_i on the l queried values it must be that f' produces different outputs for each of these values. Therefore for any $i, j \leq l, i \neq j$ it must be that $s \neq x_i \oplus x_j$. This means that there are $l(l-1)/2$ values of s which are restricted. But $l = poly(n)$ and since s can take on $2^n - 1$ possible values, if n is sufficiently large so that $2^n - 1 > l(l-1)/2$,

then we can simply choose an s which is not restricted. We therefore pick n_0 to be large enough so that $2^n - 1 > l(l-1)/2$ and then take s to be some mask from the available $2^n - 1 - l(l-1)/2$. We thus have a Simon function which produces the same responses to the queries on path π as the 1-to-1 function f_i . If we now just take f_i to be f' , then π will still be an accepting path and therefore M_k will decide incorrectly on the input $\langle 1^n, i \rangle$.

Through this construction, all non-deterministic Turing machines will have some input on which they decide $coSimon(O)$ incorrectly, thus $coSimon(O) \notin \text{NP}^O$ concluding the proof. \square

Next, we prove:

Lemma 6. *For each $d \in \mathbb{N}$, there exists an oracle O , such that $\text{BQP}^O \not\subseteq (\text{P}/\text{O}(n^d))^O$.*

Proof. To begin with, the class $\text{P}/\text{O}(n^d)$ is the class of problems solved by a deterministic polynomial-time Turing machine M , which receives an advice of length $O(n^d)$, when the input is of size $O(n)$ (in our case the input size is $2n$ since we defined n as being the length of inputs to the 1-to-1 and Simon functions).

In contrast to the previous case, instead of having the ability to non-deterministically choose one of exponentially many paths, a polynomial-time Turing machine M receives some non-uniform information to help it in deciding $coSimon(O)$. Each advice determines a new behaviour for M which can even involve a different sequence of queries to the oracle. What we want to show is that irrespective of what advice M might receive, it still cannot always correctly decide $coSimon(O)$. To do this, we consider functions over a larger domain than just n -bit strings. In other words, for each d we choose $D > d$ such that the set \mathcal{F}_n contains 2^n functions of the form $f : \{0, 1\}^{n^D} \rightarrow \{0, 1\}^{n^D}$. The oracle, which we now denote as O_d , still receives queries of the form $(1^n, i, x)$, where $|i| = n$, but now $|x| = n^D$.

First we need to argue that the problem can still be decided in BQP^{O_d} . This is indeed the case, since expanding the domains of the functions simply changes the running time of the quantum algorithm from $O(n)$ to $O(n^D)$. But since D is just a fixed constant, the algorithm still runs in polynomial time, hence $coSimon(O_d) \in \text{BQP}^{O_d}$.

The harder part is showing $coSimon(O_d) \notin \text{P}/\text{O}(n^d)$. As before, we will prove this by diagonalization by considering the set of all Turing machines (deterministic, this time) and showing that no matter which advice the k 'th machine receives it cannot correctly decide $coSimon(O_d)$. Of course, we need to be cautious as each advice induces a different behaviour and one must consider the oracle so that all of them fail to give the correct behaviour. This is in contrast with the previous case where we were only interested in the behaviour on one accepting path of the non-deterministic computation tree.

Suppose we take the k 'th deterministic polynomial-time Turing machine, M_k , and examine what happens for an input of length $n = k + n_0$, where n_0 will be chosen later (as before). Since the advice is a binary string of length $O(n^d)$ there are $2^{O(n^d)}$ possible advice strings. Whichever one M_k uses it will be the same for all 2^n inputs of length n .

Let us now consider the first index of length n , namely 0^n and assign a 1-to-1 function $f : \{0, 1\}^{n^D} \rightarrow \{0, 1\}^{n^D}$ to this index. We can inspect the behaviour of M_k for f and for each possible advice string. If for more than half of the advice strings M_k rejects, then we keep f at index 0^n . This basically means that half of all advice strings have been eliminated (there is at least one input on which those strings lead to M_k deciding incorrectly). If, however more than half of all advice strings make M_k accept f , we will attempt to turn f into a Simon function while keeping acceptance for those advices. This will again lead to the elimination of half of all advice strings.

For each advice a_j , where $1 \leq j \leq 2^{O(n^d)}$, M_k will make a sequence of polynomially many queries to f . Denote that sequence of queries together with the responses as:

$$\sigma_j = [(x_{1j}, f(x_{1j})); (x_{2j}, f(x_{2j})); \dots (x_{lj}, f(x_{lj}))]$$

where $l = \text{poly}(n)$. We now consider a Simon function $f' : \{0, 1\}^{n^D} \rightarrow \{0, 1\}^{n^D}$ such that for all j in which M_k with advice a_j and queries σ_j accepts and for all $t \leq l$, we have that $f'(x_{tj}) = f(x_{tj})$.

In other words f' will give identical responses to the queries which make M_k accept. Since t ranges from 1 to $l = \text{poly}(n)$ and j ranges from 1 to $2^{O(n^d)}$, the maximum number of variables which are queried is of order $2^{O(n^d)}$. But unlike in the previous lemma, this number is exponential in the size of the input, so how can we be sure that such a Simon function even exists? The trick is that we can choose the domain size through D and make it large enough to accommodate for a Simon function with this property.

As before, because f is bijective, no two queried variables will produce the same answer. Therefore, there cannot be a bit mask s ($|s| = n^D$) relating any pair of the $2^{O(n^d)}$ queries. These will be the restricted values of s . The total number of such values is also of order $2^{O(n^d)}$, however the total number of possible values is $2^{O(n^D)}$. So if we simply choose D such that $2^{O(n^D)} > 2^{O(n^d)}$ then we can find a Simon function f' which matches the responses of f on the $2^{O(n^d)}$ queries.

So for this case if we use f' as the function for index 0^n we will eliminate half of the possible advice strings. Thus, no matter how M_k behaves we are able to eliminate half of all possible advice strings with our first input of length $2n$. Clearly this process can be repeated for the next index and so on until the last index. We are effectively halving the number of potentially useful advice strings with each index. Since we are doing this 2^n times, to eliminate all possible advice strings we just need to ensure that $2^{O(n^d)}/2^{2^n} < 1$ or $2^{O(n^d)} < 2^{2^n}$. To achieve this, simply choose n_0 (recall that $n = k + n_0$) large enough so that the inequality holds.

We therefore have that for all k , and for all possible advice strings, there will always be an input to $\text{coSimon}(O_d)$ which is decided incorrectly, hence $\text{coSimon}(O_d) \notin \text{P/O}(n^d)$.

Note that the same proof would not work for P/poly . A crucial element in our proof was the fact that we can make D (which determines the size of the domain of each function) to be much larger than d (which determines the length of the advice). But this is only possible because d is fixed from the very beginning. If the advice length could be any arbitrary polynomial then no matter what constant value of D we decided upon for our oracle, there would always be some $d > D$ and hence some polynomial length of the advice string for which the proof does not work. A possible “fix” would be to make D part of the input in some form, so that it can increase. So if, say, D was included in the input as a $g(n)$ unary string, where g is some monotonically increasing function, then for sufficiently large n , $g(n) > d$. But we immediately notice the problem with this approach. While it is true that in this case the problem cannot be decided in P/poly^O it would also no longer be decidable in BQP^O either. This is because the query complexity of the quantum algorithm becomes $O(n^{g(n)})$ which is no longer polynomial unless g is the constant function. Hence, proving separation from P/poly seems to require some non-trivial modification of this proof or a totally different technique. \square

Finally, we can prove Theorem 2 by combining the previous two results.

Proof of Theorem 2. The oracle O_d will be defined in the exact same way as for the $\text{P/O}(n^d)$ case. We go through the same line of reasoning as before. Take the k 'th non-deterministic Turing machine and examine its behaviour for some input $\langle 1^n, i \rangle$, where $n = k + n_0$ and n_0 is chosen as before. For each index, we tentatively pick a 1-to-1 function and examine what the machine does for each advice of length $O(n^d)$. If more than half of the advices lead to rejection then we keep the bijective function and proceed to the next index. Otherwise we replace it with a Simon function. In this case, for each advice in which the machine accepts, there will be some polynomial-sized path leading to acceptance. We will pick one accepting path for each advice on which the machine accepts and ensure that the Simon function produces the same responses to the queries on those paths. This reduces the problem to the previous case. We know that for sufficiently large D such a function exists and therefore each index will render half of the possible advice strings useless. By also choosing n_0 large enough we can make sure that all advice strings are eliminated and thus that the problem is incorrectly decided by all non-deterministic Turing machines irrespective of the advice (of length $O(n^d)$). Thus $\text{coSimon}(O_d) \notin \text{NP/O}(n^d)$, concluding the proof. \square

The advantage of this proof technique is that the $\text{NP/O}(n^d)$ case reduces to the $\text{P/O}(n^d)$ case.

We therefore conjecture that if there is some modification of our proof allowing it to work for P/poly it would also work for NP/poly . Of course, this technique relies on the crucial aspect of knowing an asymptotic bound for the polynomial which determines the length of the advice. This allows us to always choose a larger polynomial for the size of the domain of the functions to be queried.

7.4 Proof of Theorem 3

Proof. As mentioned in the main text, when considering a GES for exact BOSONSAMPLING it is as if the server has the ability to query the oracle \mathcal{O} from Definition 10. The client has the description of an optical network, denoted x , and wants to obtain $\mathcal{O}(x, r)$ for a uniformly random r . As a slight abuse of notation, we denote such a sample as $\mathcal{O}(x)$. If we assume that the client can obtain $\mathcal{O}(x)$ through the GES, then there is an MA/rpoly algorithm for producing $\mathcal{O}(x)$. This follows from the same reasons as in the proof of Theorem 1. From Theorem 6 we know that $\text{MA}/\text{rpoly} = \text{MA}/\text{poly}$ so in fact there is an MA/poly algorithm for $\mathcal{O}(x)$. But the MA/poly algorithm can be viewed as an NP/poly algorithm that also receives a uniformly random string as input, which it uses as its only source of randomness. In other words, for given x and r there exists an NP/poly algorithm for computing $\mathcal{O}(x, r)$.

Combining this with the Aaronson and Arkhipov result of Theorem 7 we would have that $P^{\#P} \subset \text{BPP}^{NP^{NP}}/\text{poly}$. By Toda's theorem this means $\text{PH} \subset \text{BPP}^{NP^{NP}}/\text{poly}$ and in particular $NP^{NP^{NP}} \subset \text{BPP}^{NP^{NP}}/\text{poly}$. But Adleman's result, that $\text{BPP} \subset P/\text{poly}$ [49], relativizes and so this becomes equivalent to $NP^{NP^{NP}} \subset P^{NP^{NP}}/\text{poly}$.

Lastly, this is just a more general version of the precondition in the Karp-Lipton theorem (that if $NP \subset P/\text{poly}$ then $\Sigma_2^P \subseteq \Pi_2^P$ and PH collapses at the second level [70]) and since we know that result is also relativizing it follows that if the containment is true then $\Sigma_4^P \subseteq \Pi_4^P$ and the polynomial hierarchy collapses at the fourth level. \square

We also prove the two related lemmas from the section on sampling problems. We start with Lemma 1.

Proof of Lemma 1. First of all we need to define $\text{SampMA}/\text{poly}$. We will also be using $\text{SampMA}/\text{rpoly}$. We define both these classes as well as the “base class” SampMA :

Definition 11 (SampMA , $\text{SampMA}/\text{poly}$ and $\text{SampMA}/\text{rpoly}$). *A sampling problem P defined by the collection of distributions $(\mathcal{D}_x)_{x \in \{0,1\}^*}$ belongs to the class SampMA if there exists a probabilistic polynomial-time algorithm A (known as Arthur) and a deterministic polynomial-time algorithm V (known as verification algorithm) such that the following is true for all $x \in \{0,1\}^*$:*

- *If there exists a $y \in \{0,1\}^{\text{poly}(|x|)}$ such that V accepts on input $\langle x, y \rangle$, then A on input $\langle x, y, 0^{1/\epsilon} \rangle$, produces a sample from some \mathcal{C}_x such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \epsilon$.*
- *If for all $y \in \{0,1\}^{\text{poly}(|x|)}$ V rejects on input $\langle x, y \rangle$, then A on input $\langle x, y, 0^{1/\epsilon} \rangle$, produces a sample from some \mathcal{C}_x such that $\|\mathcal{C}_x - \mathcal{D}_x\| \geq \epsilon$.*

For the class $\text{SampMA}/\text{poly}$, A and V will also receive as input an advice string $s \in \{0,1\}^{\text{poly}(|x|)}$ which only depends on the length of x . Lastly, for $\text{SampMA}/\text{rpoly}$ V is a probabilistic polynomial-time algorithm and A produces the correct sample when V accepts with probability greater than $2/3$ and an incorrect sample when V accepts with probability less than $1/3$. Furthermore, both A and V receive as input an advice string $s \in \{0,1\}^{\text{poly}(|x|)}$, drawn from a distribution $\mathcal{D}_{|x|}$ that only depends on the size of the input x .

Similar to the proof of Theorem 1, for a problem that admits a GES it is easier to define an algorithm that receives randomized advice. Therefore, we will give a $\text{SampMA}/\text{rpoly}$ algorithm for sampling problems P that admit a GES. To then show that $P \in \text{SampMA}/\text{poly}$ we need to first prove that $\text{SampMA}/\text{poly} = \text{SampMA}/\text{rpoly}$.

Lemma 7. $\text{SampMA/poly} = \text{SampMA/rpoly}$

Proof. In the decision case we already know that $\text{MA/poly} = \text{MA/rpoly} = \text{NP/poly}$ (from Theorem 6), but the equality among the corresponding sampling problems does not follow immediately. This is because, when dealing with decision problems, for each input one only needs to decide whether it belongs to a certain language or not. The proof is then a derandomization argument whereby one can show that if a randomized algorithm (with randomized advice) can decide a problem with bounded error, then there is some deterministic advice which will make the algorithm always decide correctly. In the case of sampling classes, however, the problems themselves are not deterministic yes/no problems. Hence, we can never have a sampling algorithm that does not utilize randomness. What we can show, however, is that the only randomness which is necessary comes from the coin flips that Arthur performs and that the advice can be made deterministic. We do this in a manner similar to the proof that $\text{MA/poly} = \text{MA/rpoly}$, from [64], by boosting and derandomizing the verification circuit of a SampMA/rpoly algorithm.

Clearly $\text{SampMA/poly} \subseteq \text{SampMA/rpoly}$, since the randomized advice can simply be drawn from a distribution peaked at a single point. We show the other direction. Consider a SampMA/rpoly instance consisting of the tuple of algorithms (V, A) as per Definition 11. We know that V is a BPP/rpoly machine acting on $z = \langle x, y, r \rangle$, where x is the input to the sampling problem, y is the witness to be verified and r is the randomized advice. We are also assuming that $0^{1/\varepsilon}$ is part of this input but ignore it for simplicity, as it does not affect the proof. Whenever A produces a correct sample, $V(z)$ accepts with probability greater than $2/3$ and whenever A produces an incorrect sample, $V(z)$ accepts with probability less than $1/3$. The probabilities, in this case, are over the random string $r \in \mathcal{D}_{|x|}$, for some distribution $\mathcal{D}_{|x|}$. As in the proof of Theorem 6 from [64], define $R = (r_1, r_2 \dots r_{p(|x|)})$ to be a collection of independent samples from $\mathcal{D}_{|x|}$, such that $p(|x|) = |x| + \text{poly}(|x|)$. Then, there exists a boosted verifier V' such that, whenever A produces a correct sample, $V'(\langle x, y, R \rangle)$ accepts with probability greater than $1 - 1/(2^{|x| + \text{poly}(|x|)})$ and whenever A produces an incorrect sample $V'(\langle x, y, R \rangle)$ accepts with probability less than $1/(2^{|x| + \text{poly}(|x|)})$. Therefore, by a counting argument, there exists a fixed advice string r' such that whenever A produces a correct sample $V(\langle x, y, r' \rangle)$ accepts and whenever B produces an incorrect sample $V(\langle x, y, r' \rangle)$ rejects. In other words, V is a P/poly algorithm. Hence, there exists a SampMA/poly algorithm for each problem in SampMA/rpoly .

We have thus shown that $\text{SampMA/poly} = \text{SampMA/rpoly}$. \square

Returning to our proof of Lemma 1, suppose now that we have a GES for some sampling problem P consisting of the distributions $(\mathcal{D}_x)_{x \in \{0,1\}^*}$ as per Definition 6. Throughout the proof we assume that ε is known in advance to both the client and the server. We can define a SampMA/rpoly algorithm (and hence a SampMA/poly algorithm) for P just like in the proof of Theorem 1. In other words, consider a one round GES for P :

1. The client runs $K(x)$ until success to produce an encryption key k .
2. The client computes the encrypted string $y \leftarrow E(x, k, '')$ (where the last entry is the empty string) and sends it to the server.
3. The server sends a response r .
4. The client decrypts his response obtaining $z \leftarrow D(r, k, x)$ such that z is a sample from \mathcal{C}_x and $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$.

To show that $P \in \text{SampMA/rpoly}$ we need to define the two algorithms A and V . We start with the verification algorithm, V . The input to V is x , the witness we denote as k and the advice will consist of 3 strings of length $n = |x|$, namely $\langle x_n, y_n, r_n \rangle$. V ignores x_n and r_n and accepts iff $y_n \leftarrow E(x, k, '')$. Essentially V is just running the encryption algorithm with input x and key k and checking to see whether it equals the middle part of the advice string, y_n . We now define A :

- Again, denoting $|x| = n$, the algorithm receives as advice some string $x_n \in \{0,1\}^n$, $y_n \leftarrow E(x_n, k_n, \cdot)$, where k_n is simply some key which can be used to encrypt x_n , as well as r_n , where r_n is the server's response when being sent y_n . The string x_n is included to check whether $x_n = x$. If this is the case then the algorithm simply decrypts r_n obtaining a sample from \mathcal{C}_x and $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$. The next steps assume that $x_n \neq x$.
- From the assumption that the GES leaks at most the size of the input, there must be some key k , such that $y_n \leftarrow E(x, k, \cdot)$. This is because if there did not exist such a key, then if the server received y_n he would know that the input could not be x and hence more information would be leaked. In other words the verification algorithm V accepts when given as input x , witness k and advice containing y_n .
- With the key k that is accepted by V , A can simply compute $z \leftarrow D(r_n, k, x)$ and z will be a sample from \mathcal{C}_x such that $\|\mathcal{C}_x - \mathcal{D}_x\| \leq \varepsilon$.

Thus, any sampling problem P which admits a GES leaking at most the size of the index specifying the distribution is contained in $\text{SampMA}/\text{rpoly} = \text{SampMA}/\text{poly}$. This concludes the proof. \square

Lastly we give the proof for Lemma 2.

Proof of Lemma 2. This is simply a combination of the previous two results. If $\text{BOSONSAMPLING} \in \text{SampMA}/\text{poly}$ and Conjecture 1 is true (estimating the permanent of a Gaussian matrix is $\#\text{P}$ -complete), then just like in Theorem 3 we would have that $\text{P}^{\#\text{P}} \subset \text{BPP}^{\text{NP}^{\text{NP}}}/\text{poly}$. But we have already shown that this would lead to a collapse of the polynomial hierarchy at the fourth level. Hence, since $\text{BOSONSAMPLING} \in \text{SampBQP}$ we have that $\text{SampBQP} \subset \text{SampMA}/\text{poly}$ leads to a collapse of the polynomial hierarchy. \square

7.5 Proof of Lemma 3

Proof. To show that UBQC is a type of QGES we only need to give implementations for the algorithms K , QE , E and D which are consistent with UBQC and the properties of a QGES leaking at most the size of the input.

- Key generation, K . This is the step in which the client chooses the random angles for the $|+\theta\rangle$ states that it will send to the server as well as the bits for randomly flipping the measurement outcomes. Thus, K simply takes as input x and produces $M = \text{poly}(|x|)$ random angles $\langle\theta_1, \theta_2, \dots, \theta_M\rangle$ drawn at random from the set $\{0, \pi/4, 2\pi/4, \dots, 7\pi/4\}$ and random bits $\langle r_1, r_2, \dots, r_M\rangle$. Thus the classical key is $k = \{\langle\theta_1, \theta_2, \dots, \theta_M\rangle, \langle r_1, r_2, \dots, r_M\rangle\}$.
- Quantum encryption, QE . In this step the client uses to key to prepare the qubits that it will send to the server. In other words $QE(x, k) \rightarrow |+\theta_1\rangle |+\theta_2\rangle \dots |+\theta_M\rangle$.
- Computation, E . In round i , the output of E will be the angles $\Delta_i = \langle\delta_{i,1}, \delta_{i,2}, \dots, \delta_{i,k}\rangle$ which the server should use to measure the qubits in layer i . These are computed based on x , k and the result of the server's previous responses. Concretely, for a particular qubit j , E will compute $\delta_j = (-1)^{r_j} \phi_j + \theta_j + r'_j \pi$, where ϕ_j is the computation angle (in part determined by x), θ_j is the randomization of the measurement and is contained in the key k , and r'_j is the randomization of the measurement outcomes (computed by xor'ing previous measurement outcomes and the random parameter r_j , contained in k).
- Decryption, D . The decryption procedure simply involves xoring previous measurement outcomes and the random parameter r_j for the output qubit to determine its true value.

Since this is a BQP computation, the probability of obtaining the correct outcome will be at least $2/3 > 1/2 + 1/\text{poly}(|x|)$. This shows that UBQC is a QGES. Additionally, since we already know that UBQC leaks at most the size of x to the server, this particular QGES leaks the same information. \square

7.6 Proof of Theorem 4

Proof. For an input x for which the client wants to compute $f(x)$, consider the state:

$$|\psi_x\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x)|}} \sum_{k_i^x \in \mathcal{K}_C(x)} |k_i^x\rangle_K |y_i^x\rangle_E \quad (9)$$

Where $\mathcal{K}_C(x)$ is the set of encryption keys which are compatible with x (i.e. could have resulted from the key generation algorithm acting on x , $K(x)$) and $|y_i^x\rangle_E \leftarrow QE(x, k_i^x)$ is the quantum encryption of x using the key k_i^x . The indices K and E specify whether the kets are quantum registers in the key register or the encrypted state register, respectively. Essentially $|\psi_x\rangle$ is the equal superposition of all keys and encryptions of the string x . If we trace out the key register, K , the resulting density matrix is the mixed state of possible encrypted states which the server will receive:

$$\rho_x = \frac{1}{|\mathcal{K}_C(x)|} \sum_{k_i^x \in \mathcal{K}_C(x)} |y_i^x\rangle \langle y_i^x| \quad (10)$$

The assumption that the protocol only leaks the size of the input x to the server implies that for any two inputs x_1, x_2 it is the case that $\rho_{x_1} = \rho_{x_2}$. In fact, something stronger is true. Recall that the definition of blindness says that the quantum state of the server's system as well as the distribution of his classical messages are independent of x , given the size of x . Therefore, we should consider a state comprising of his system and his response after receiving the quantum encryption, for a particular input, x :

$$|\phi_x\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x)|}} \sum_{k_i^x \in \mathcal{K}_C(x)} |k_i^x\rangle_K U_{ERS} |y_i^x\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S \quad (11)$$

Here, U_{ERS} is the unitary performed by the server in order to produce his response, which will be stored in the response register, initially set to $|0\rangle_R^{\otimes t}$, where $t = \text{poly}(|x|)$. This unitary will of course involve the encrypted state provided by the client and the server's private ancilla, denoted as $|anc\rangle_S$ (but will not involve the key register). Note that in the actual protocol, the key register and the encrypted state register are not necessarily entangled. For example, in UBQC they are only classically correlated. However, since we are considering the most general case, we take the state to be entangled. Essentially the client's system can be thought of as a purification system for the encrypted quantum state sent to the server. Additionally, it should be mentioned that the server's response is a classical bit string. Hence, the state in the response register, obtained through the application of the unitary U_{ERS} , will be a probabilistic mixture over computational basis states. This, however, makes no difference in our proof and we can just as well assume that his response is a general quantum state.

If we again trace out the register K we obtain some state $\sigma_x = \text{Tr}_K(|\phi_x\rangle \langle \phi_x|)$. This state encodes the distribution of possible messages exchanged by the client and the server in one round of interaction, as well as the server's private system. Since $\rho_{x_1} = \rho_{x_2}$, it is also the case that $\sigma_{x_1} = \sigma_{x_2}$. This is exactly the blindness condition. By the purification principle (see Appendix 7.1.1), this means that if we consider the states:

$$|\phi_{x_1}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x_1)|}} \sum_{k_i^{x_1} \in \mathcal{K}_C(x_1)} |k_i^{x_1}\rangle_K U_{ERS} |y_i^{x_1}\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S \quad (12)$$

$$|\phi_{x_2}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x_2)|}} \sum_{k_i^{x_2} \in \mathcal{K}_C(x_2)} |k_i^{x_2}\rangle_K U_{ERS} |y_i^{x_2}\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S \quad (13)$$

there exists a local unitary, V_K , acting only on the key register which can map $|\phi_{x_1}\rangle$ to $|\phi_{x_2}\rangle$, for any two inputs x_1 and x_2 . In fact, let us examine the states of the system for inputs x_1 and x_2

before U_{ERS} is applied:

$$|\chi_{x_1}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x_1)|}} \sum_{k_i^{x_1} \in \mathcal{K}_C(x_1)} |k_i^{x_1}\rangle_K |y_i^{x_1}\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S = |\psi_{x_1}\rangle_{KE} |0\rangle_R^{\otimes t} |anc\rangle_S \quad (14)$$

$$|\chi_{x_2}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x_2)|}} \sum_{k_i^{x_2} \in \mathcal{K}_C(x_2)} |k_i^{x_2}\rangle_K |y_i^{x_2}\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S = |\psi_{x_2}\rangle_{KE} |0\rangle_R^{\otimes t} |anc\rangle_S \quad (15)$$

These states are also related by V_K . This can be inferred from the following relations. First, we know that:

$$(V_K \otimes I_{ERS}) |\phi_{x_1}\rangle = |\phi_{x_2}\rangle \quad (16)$$

And also that:

$$(I_K \otimes U_{ERS}) |\chi_{x_1}\rangle = |\phi_{x_1}\rangle \quad (I_K \otimes U_{ERS}) |\chi_{x_2}\rangle = |\phi_{x_2}\rangle \quad (17)$$

Therefore:

$$(I_K \otimes U_{ERS}^\dagger)(V_K \otimes I_{ERS})(I_K \otimes U_{ERS}) |\chi_{x_1}\rangle = |\chi_{x_2}\rangle \quad (18)$$

But $(I_K \otimes U_{ERS}^\dagger)$ and $V_K \otimes I_{ERS}$ commute because they act on different systems and therefore $(I_K \otimes U_{ERS}^\dagger)$ and $(I_K \otimes U_{ERS})$ will cancel out, leaving:

$$(V_K \otimes I_{ERS}) |\chi_{x_1}\rangle = |\chi_{x_2}\rangle \quad (19)$$

This is also illustrated in the following diagram:

$$\begin{array}{ccc} |\chi_{x_1}\rangle & \xrightarrow{V_K \otimes I_{ERS}} & |\chi_{x_2}\rangle \\ \downarrow I_K \otimes U_{ERS} & & \uparrow I_K \otimes U_{ERS}^\dagger \\ |\phi_{x_1}\rangle & \xrightarrow{V_K \otimes I_{ERS}} & |\phi_{x_2}\rangle \end{array}$$

But because the protocol is offline, we know that V_K must be a polynomial-sized quantum circuit. Note that even if we trace out the server's ancilla from the states $|\phi_{x_1}\rangle$ and $|\phi_{x_2}\rangle$, the resulting states are still related by V_K on the key register. This allows us to define a QCMA/qpoly algorithm computing any function which admits a QGES. To do so we first introduce some notation. We will consider the following states:

$$|\kappa_x\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x)|}} \sum_{k_i^x \in \mathcal{K}_C(x)} |k_i^x\rangle_K \quad (20)$$

$$|\kappa_{x'}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x')|}} \sum_{k_i^{x'} \in \mathcal{K}_C(x')} |k_i^{x'}\rangle_K \quad (21)$$

Which are simply superpositions over the valid keys for two different inputs x and x' . Next we consider:

$$|\phi_x\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x)|}} \sum_{k_i^x \in \mathcal{K}_C(x)} |k_i^x\rangle_K U_{ERS} |y_i^x\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S \quad (22)$$

$$|\phi_{x'}\rangle = \frac{1}{\sqrt{|\mathcal{K}_C(x')|}} \sum_{k_i^{x'} \in \mathcal{K}_C(x')} |k_i^{x'}\rangle_K U_{ERS} |y_i^{x'}\rangle_E |0\rangle_R^{\otimes t} |anc\rangle_S \quad (23)$$

which include the encrypted states and the server's response. Lastly, we trace out the server's ancilla from both these states resulting in:

$$\omega_x = \text{Tr}_S(|\phi_x\rangle\langle\phi_x|) \quad (24)$$

$$\omega_{x'} = \text{Tr}_S(|\phi_{x'}\rangle\langle\phi_{x'}|) \quad (25)$$

From the above argument the two states $|\kappa_x\rangle$ and $|\kappa_{x'}\rangle$ and the two states ω_x and $\omega_{x'}$ are related through the same polynomial-sized quantum circuit V_K acting only on the key register.

We can now present the algorithm. Let us first consider the one round case. The algorithm would work as follows:

1. The input to the algorithm is some string x for which we want to compute $f(x)$.
2. The algorithm receives as advice the string x' which is simply some string of the same length as x . Additionally, it receives the state $\omega_{x'}$. It is clear that both of these only depend on $|x|$ and have a length which is polynomial in $|x|$ hence constituting a valid advice.
3. From the definition of the key generating function, the algorithm can efficiently produce the states $|\kappa_x\rangle$ and $|\kappa_{x'}\rangle$.
4. The classical witness is a description of the quantum circuit V_K^\dagger .
5. The algorithm tests that V_K^\dagger maps $|\kappa_{x'}\rangle$ to $|\kappa_x\rangle$. This can be done through a quantum SWAP test.
6. Use V_K^\dagger to map $\omega_{x'}$ to ω_x .
7. By measuring the response register of ω_x , the algorithm obtains the response that the server would have produced in an interaction with the client in the QGES protocol. Applying the decryption algorithm to this response will yield the correct result $f(x)$ with high probability.

The probability of success of the algorithm can be boosted by providing polynomially many copies of $\omega_{x'}$ as advice and performing multiple SWAP tests. Additionally this algorithm can be made to compute the complement of $f(x)$ as well which would give us a **coQCMA/qpoly** containment.

For the general case of polynomially many rounds, the only difference is that the state $\omega_{x'}$ would also be entangled with a superposition of all possible transcripts of the protocol. Since we know that transcripts are polynomially bounded in length this is still a valid advice state. The application of V_K^\dagger would map this state to one containing the transcripts for input x . When the state is measured the algorithm will obtain a sample transcript of the interaction between the client and the server. This is then used together with the decryption algorithm to produce $f(x)$. By the definition of the QGES we know that the possible transcripts are such that the correct $f(x)$ is obtained with high probability.

Note that depending on how we define the offline property of the protocol we can get containments in different classes. For example, in this proof we have assumed that while there is a polynomial-sized circuit allowing the client to map from one input to another the client might not be able to arrive at this circuit in polynomial time for any possible pair of inputs. This is why the description of the circuit is given as a witness (since the client can always test the validity of this circuit). However, if we additionally assumed that V_K^\dagger can always be obtained efficiently by the client, then we would no longer need the witness and we would have that $f \in \text{BQP/qpoly}$. \square

We also prove that:

Lemma 8. $\text{BQP}^{\text{QCMA/qpoly} \cap \text{coQCMA/qpoly}} = \text{QCMA/qpoly} \cap \text{coQCMA/qpoly}$

Proof. This proof is similar to the one showing that $\text{BPP}^{\text{NP}/\text{poly}} \cap \text{coNP}/\text{poly} = \text{NP}/\text{poly} \cap \text{coNP}/\text{poly}$. Just like in that case, the inclusion $\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly} \subseteq \text{BQP}^{\text{QCMA}/\text{qpoly}} \cap \text{coQCMA}/\text{qpoly}$ is immediate and we need only show that $\text{BQP}^{\text{QCMA}/\text{qpoly}} \cap \text{coQCMA}/\text{qpoly} \subseteq \text{QCMA}/\text{qpoly}$. The containment in $\text{coQCMA}/\text{qpoly}$ follows by complementation.

Consider a quantum algorithm QA for deciding problems in $\text{BQP}^{\text{QCMA}/\text{qpoly}} \cap \text{coQCMA}/\text{qpoly}$. We will show that this algorithm can be simulated by a QCMA/qpoly algorithm, denoted NQA . Since BQP , QCMA and coQCMA have bounded error in deciding problems, we can assume, from standard amplification techniques, that this error is of order $2^{-\text{poly}(n)}$, where n is the size of the input. We will also assume that for all quantum algorithms measurements are postponed until the end of the circuit.

We will treat the case without advice first, and then explain how to deal with the quantum advice at the end. To start with, NQA will simulate QA until it makes a query to the oracle. In the standard definition of oracles the oracle is just a classical function that solves a decision problem. However, when dealing with quantum algorithms such as QA it is also possible to speak of quantum oracles, where the oracle can be viewed as some unitary operation (technically a sequence of unitary operations for each possible input length, see [71] for more details) which QA can query even in superposition. Our result will cover this more general case of quantum oracles. We would therefore like the NQA algorithm to be able to simulate this quantum oracle.

Firstly, just like in the classical case we have that if some language $L \in \text{QCMA} \cap \text{coQCMA}$ then $L \in \text{QCMA}$ and $L^c \in \text{coQCMA}$, where L^c is the complement of L . This means that there exist polynomial-sized quantum circuits Q_L and Q_{L^c} which take some input x along with classical witnesses w_1 and w_2 , respectively, and decide correctly, when the output is measured, with probability at least, $1 - 2^{-\text{poly}(|x|)}$. In other words, Q_L receives as input $|x\rangle |w_1\rangle |0^m\rangle$ and Q_{L^c} receives as input $|x\rangle |w_2\rangle |0^m\rangle$, respectively, where $m = \text{poly}(|x|)$. If we were to run both Q_L and Q_{L^c} on x , because L and L^c are complementary, the output qubits, when measured, will also be complementary with high probability.

Assume that Q_L and Q_{L^c} are circuits which act on $t = \text{poly}(|x|)$ qubits. We define a new quantum circuit called *SimQuery* which operates on $2t + 1$ qubits. *SimQuery* applies Q_L to the first t qubits and Q_{L^c} to the next t qubits. It then applies a Pauli X to the output qubit of Q_{L^c} and a CCNOT operation from the output qubits of Q_L and Q_{L^c} onto the $2t + 1$ 'th qubit. It then applies X again to the output qubit of Q_{L^c} and then Q_L^\dagger and $Q_{L^c}^\dagger$ on the first $2m$ qubits. An illustration of this circuit (acting on an all $|0\rangle$ input) is given in Figure 6.

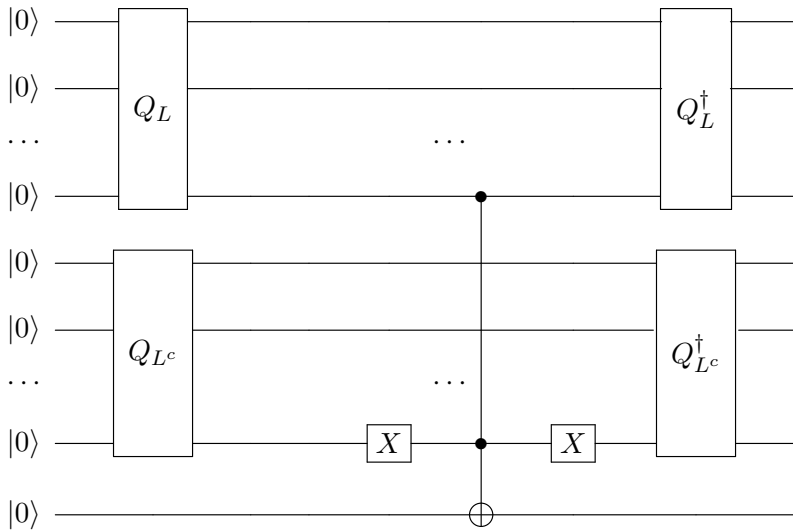


Figure 6: Quantum circuit for *SimQuery* acting on an all $|0\rangle$ input.

The CCNOT operation flips its target qubit if the control qubits are in the state $|11\rangle$. The effect of the first Pauli X is to flip the outcome when the control qubits are in the state $|10\rangle$. Roughly

speaking, *SimQuery* will flip the final qubit if Q_L accepts and Q_{L^c} rejects. The reason for then applying the two circuits in reverse is to ‘uncompute’ their result and only leave the $2t + 1$ qubit flipped whenever Q_L accepts and Q_{L^c} rejects.

We can now explain how *NQA* can use *SimQuery* to simulate a query of *QA*. Suppose *QA* queries the oracle for some input x testing to see if it is in L , for some $L \in \text{QCMA} \cap \text{coQCMA}$. In other words, $|x\rangle|0\rangle \rightarrow |x\rangle|1\rangle$, with high probability, if $x \in L$ and $|x\rangle|0\rangle \rightarrow |x\rangle|0\rangle$, with high probability, if $x \notin L$. *NQA* will then run *SimQuery* with input $|x\rangle|w_1\rangle|0^m\rangle|x\rangle|w_2\rangle|0^m\rangle|0\rangle$, where w_1 and w_2 are the witnesses from before and $m = \text{poly}(|x|)$. The effect of this will be to flip the final qubit if $x \in L$ and leave it unchanged if $x \notin L$, with high probability. This is true because of the complementarity of Q_L and Q_{L^c} (when one accepts the other rejects and viceversa, except with small probability).

This procedure will simulate the query that *QA* performs. *NQA* then uses the last qubit from *SimQuery* as the query response qubit and continues to do this for all other queries of *QA* and otherwise simulate *QA* exactly. Note that each simulated query has some small probability of not matching the actual query of *QA* when a measurement is performed. However, as mentioned, this probability is exponentially small. Since there are polynomially many queries in total, by a union bound, the probability that at least one simulated query behaves incorrectly will still be exponentially small.

Adding quantum advice to this picture does not change much. Just like in the classical case, we can assume that *NQA* receives as advice a concatenation of all advice states used by the oracle of *QA*. The quantum circuits Q_L , Q_{L^c} and *SimQuery* are then extended with polynomially many qubits to act on this advice as well.

It is therefore the case that $\text{BQP}^{\text{QCMA}/\text{qpoly}} \cap \text{coQCMA}/\text{qpoly} \subseteq \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$ and our result follows directly. \square

7.7 Proof of Theorem 5

In this appendix, we prove Theorem 5. Since we have shown that functions which admit an offline QGES are contained in $\text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$, and since if $\text{NP} \subset \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$ then $\text{coNP} \subset \text{QCMA}/\text{qpoly} \cap \text{coQCMA}/\text{qpoly}$, to prove Theorem 5, it suffices to show that if $\text{coNP} \subset \text{QCMA}/\text{qpoly}$ then the polynomial hierarchy “comes about as close to collapsing as one could reasonably hope to prove given a quantum hypothesis”—and more specifically, that $\Pi_3^P \subseteq (\Sigma_2^P)^{\text{PromiseQMA}}$. Here a PromiseQMA oracle means an oracle for some PromiseQMA-complete promise problem $(\Pi_{\text{YES}}, \Pi_{\text{NO}})$, whose responses can be arbitrary on inputs $x \notin \Pi_{\text{YES}} \cup \Pi_{\text{NO}}$ that violate the promise. We don’t even demand that the oracle’s responses, on promise-violating inputs, be consistent from one query to the next. On the other hand, it does need to be possible to query the PromiseQMA oracle on some promise-violating inputs, without such queries causing the entire algorithm to abort.

The starting point for all such collapse results, of course, is the Karp-Lipton Theorem [70], which says that if $\text{NP} \subset \text{P}/\text{poly}$ then $\Pi_2^P \subseteq \Sigma_2^P$, and hence the polynomial hierarchy collapses to the second level. An easy extension of the Karp-Lipton theorem, proved by Yap [53], which we now reprove for completeness, shows that if $\text{coNP} \subset \text{NP}/\text{poly}$, then PH collapses to the *third* level.

Proposition 1. *If $\text{coNP} \subset \text{NP}/\text{poly}$, then $\Pi_3^P \subseteq \Sigma_3^P$.*

Proof. Abusing notation, here and later in this appendix we’ll use Φ , Ψ , etc. to refer not only to SAT instances but to strings encoding those instances. Also, if (say) $\Phi(x, y, z)$ is a SAT instance taking multiple strings as input, then by $\Phi(x, y)$, we’ll mean the instance obtained from Φ by fixing the variables in x and y , and leaving only the variables in z as free variables.

A Π_3^P sentence has the form

$$S = “\forall x \exists y \forall z \Phi(x, y, z)”$$

where x, y, z are strings of some given polynomial size, and Φ is a polynomial-time computable predicate (without loss of generality, a SAT instance). Under the stated hypothesis, we need to show how to decide S in Σ_3^P .

Let C be the assumed NP/poly algorithm for coNP , and let a be its advice. Then by hypothesis, for all SAT instances Ψ , if Ψ is unsatisfiable then there exists a witness w such that $C(\Psi, w, a)$ accepts, while if Ψ is satisfiable then $C(\Psi, w, a)$ rejects for all w .

Our Σ_3^P rewriting of S is now as follows:

There exists an advice string a , such that

- (1) (**Completeness of C**) For all SAT instances Ψ , either there exists a z that satisfies Ψ , or else there exists a w such that $C(\Psi, w, a)$ accepts.
- (2) (**Soundness of C**) For all SAT instances Ψ , all satisfying assignments z for Ψ , and all w , the procedure $C(\Psi, w, a)$ rejects.
- (3) (**Truth of S**) For all x , there exists a y as well as a witness w such that $C(\neg\Phi(x, y), w, a)$ accepts. (In other words, there is no z that makes $\Phi(x, y, z)$ false.)

□

Proposition 1 is what we seek to imitate in the quantum setting, getting whatever leverage we can from the weaker assumption $\text{coNP} \subset \text{QCMA}/\text{poly}$.

Note that, had we assumed (say) $\text{coNP} \subset \text{QCMA}/\text{poly}$, it would be routine to mimic the usual Karp-Lipton argument, merely substituting the class PromiseQCMA for NP at appropriate points in the proof of Proposition 1. This would give us the collapse $\Pi_3^P \subseteq (\Sigma_2^P)^{\text{PromiseQCMA}}$. However, the fundamental difficulty we face is that our hypothesized nonuniform algorithm uses *quantum advice states*. And while a PromiseQMA machine can simply guess a quantum advice state σ , it can't then pass σ to an oracle, at least not with conventional oracle calls. (To allow the passing of quantum states to oracles, we would need *quantum oracles*, as studied for example by Aaronson and Kuperberg [71].)

To get around this difficulty, we'll rely essentially on a 2010 result of Aaronson and Drucker [58, 72], characterizing the power of quantum advice. These authors proved that BQP/qpoly is contained in QMA/poly —and even more strongly,

Theorem 9 (Aaronson-Drucker [72]). $\text{BQP}/\text{qpoly} = \text{YQP}/\text{poly}$.

Here YQP , known as Yoda quantum polynomial-time, is the class of problems solvable by a polynomial-time quantum algorithm with help from a polynomial-size *untrusted* quantum advice state that depends only on the input length n . In other words, Theorem 9 says that we can simulate trusted quantum advice by trusted classical advice combined with untrusted quantum advice, by using the classical advice to verify the quantum advice for ourselves.

By using Theorem 9, to replace a quantification over quantum advice states by a quantification over classical advice strings, Aaronson and Drucker were able to show the following:

Theorem 10 (Aaronson-Drucker [72]). *If $\text{NP} \subset \text{BQP}/\text{qpoly}$, $\Pi_2^P \subseteq \text{QMA}^{\text{PromiseQMA}}$.*

By adapting our argument from later in this appendix, one can actually improve Theorem 10, to show that if $\text{NP} \subset \text{BQP}/\text{qpoly}$ then $\Pi_2^P \subseteq \text{NP}^{\text{PromiseQMA}}$. In any case, we now seek a common generalization of the proofs of Proposition 1 and Theorem 10, to get a collapse from the assumption $\text{coNP} \subset \text{QCMA}/\text{poly}$.

As Aaronson and Drucker [72] pointed out, a simple extension of their proof of Theorem 9 gives $\text{QCMA}/\text{qpoly} \subseteq \text{QMA}/\text{poly}$, and even the following.

Theorem 11 (Aaronson-Drucker [72]). $\text{QCMA}/\text{qpoly} = \text{YQ} \cdot \text{QCMA}/\text{poly}$.

Here the $\text{YQ} \cdot$ operator simply adds untrusted quantum advice to whatever (quantum) complexity class it acts on. Thus $\text{YQ} \cdot \text{BQP} = \text{YQP}$, while for completeness:

Definition 1. $\text{YQ} \cdot \text{QCMA}$ is the class of languages L for which there exist polynomial-time quantum algorithms C and V , such that for all input lengths n :

- There exists a polynomial-size quantum advice state σ_n such that $V(0^n, \sigma_n)$ accepts with probability at least 0.99. If $V(0^n, \sigma)$ accepts with probability at least 0.98, then we call the advice state σ “valid” for input length n .
- For all inputs $x \in \{0, 1\}^n \cap L$ and all valid σ , there exists a polynomial-size classical witness w such that $C(x, w, \sigma)$ accepts with probability at least $2/3$.
- For all inputs $x \in \{0, 1\}^n \setminus L$, all classical witnesses w , and all valid σ , we have that $C(x, w, \sigma)$ accepts with probability at most $1/3$.

In what follows, we’ll need one additional observation about the proof of Theorem 11. Namely, in our $\text{YQ} \cdot \text{QCMA}/\text{poly}$ simulation of QCMA/qpoly , without loss of generality we can choose the classical advice string $a = a_n$ in such a way that there’s essentially just *one* valid quantum advice state compatible with a . Or more precisely: we can ensure that, for all ρ_1, ρ_2 such that $V(0^n, a, \rho_1)$ and $V(0^n, a, \rho_2)$ both accept with probability at least 0.98, and all x and w , we have (say)

$$|\Pr[C(x, w, a, \rho_1) \text{ accepts}] - \Pr[C(x, w, a, \rho_2) \text{ accepts}]| < \frac{1}{20}.$$

This is because Theorem 11, like Theorem 9, is proven via the method of “majority-certificates,” in which given a polynomial-time quantum algorithm Q , one verifies that an unknown quantum state ρ leads to approximately the desired values of $\Pr[Q(x, \rho) \text{ accepts}]$ for *each* of exponentially many different inputs x , via a measurement of ρ that takes only polynomial time. We note that this works only because of special structure in ρ —but for any state σ , there exists another state ρ that has the requisite special structure, as well as a modified quantum algorithm Q' , such that

$$\Pr[Q'(x, \rho) \text{ accepts}] \approx \Pr[Q(x, \sigma) \text{ accepts}]$$

for all x .

We’re finally ready to prove Theorem 5.

Theorem 12. Suppose $\text{coNP} \subset \text{QCMA}/\text{qpoly}$. Then $\Pi_3^P \subseteq (\Sigma_2^P)^{\text{PromiseQMA}}$.

Proof. A Π_3^P sentence has the form

$$S = “\forall x \exists y \forall z \Phi(x, y, z)”$$

where x, y, z are strings of some given polynomial size, and Φ is a polynomial-time computable predicate. Under the stated hypothesis, we need to show how to decide S in $\text{NP}^{\text{NP}^{\text{PromiseQMA}}}$.

By Theorem 11, the hypothesis $\text{coNP} \subset \text{QCMA}/\text{qpoly}$ is equivalent to $\text{coNP} \subset \text{YQ} \cdot \text{QCMA}/\text{poly}$. In other words: we can assume that there exists a polynomial-time quantum algorithm $C(\Phi, w, a, \sigma)$, which takes as input a SAT instance Φ , a classical witness w , a classical advice string a , and a quantum advice state σ . Assuming a and σ are the correct $\text{YQ} \cdot \text{QCMA}/\text{poly}$ advice, C checks whether w is a witness to Φ ’s unsatisfiability. This is a sound and complete proof system for coNP , in the sense that, again assuming the correctness of a and σ ,

- for every unsatisfiable Φ , there exists a w such that $C(\Phi, w, a, \sigma)$ accepts with probability at least $2/3$,
- for no satisfiable Φ does there exist a w such that $C(\Phi, w, a, \sigma)$ accepts with probability more than $1/3$.

Moreover, as discussed above, there exists an a such that the state σ is essentially unique, in the sense that

$$\Pr [C(\Psi, w, a, \rho_1) \text{ accepts}] \approx \Pr [C(\Psi, w, a, \rho_2) \text{ accepts}]$$

for all valid ρ_1, ρ_2 .

Our job is to rewrite S as an $\text{NP}^{\text{NP}^{\text{PromiseQMA}}}$ sentence. Our rewriting will be as follows:

There exists a classical advice string a such that

- (1) for all valid quantum advice states ρ_1, ρ_2 , all SAT instances Ψ , and all assignments w , we have

$$|\Pr [C(\Psi, w, a, \rho_1) \text{ accepts}] - \Pr [C(\Psi, w, a, \rho_2) \text{ accepts}]| < \frac{1}{10}.$$

(In words: the classical advice string a uniquely determines the behavior of C , once we find a valid quantum advice state σ that's compatible with a .)

- (2) For all SAT instances Ψ , there exists a valid quantum advice state σ , as well as either an assignment z that satisfies Ψ , or else a classical witness w such that $C(\Psi, w, a, \sigma)$ accepts with probability at least $2/3$.

(In words: the advice a leads to a complete procedure for deciding the class coNP , and specifically the UNSAT problem, in $\text{YQ} \cdot \text{QCMA}/\text{poly}$. That is, once we find a valid advice state σ , the quantum algorithm C then accepts every SAT instance Ψ that has no satisfying assignment.)

- (3) For all valid quantum advice states σ , all SAT instances Ψ , all z , and all w , if z satisfies Ψ then $C(\Psi, w, a, \sigma)$ rejects with probability at least $2/3$.

(In words: a leads to a *sound* procedure for deciding UNSAT. That is, once we find a valid σ that's compatible with a , the quantum algorithm C accepts no SAT instance Ψ that *has* a satisfying assignment.)

- (4) For all x , there exists a valid quantum advice state σ , as well as a y and a classical witness w , such that $C(\neg\Phi(x, y), w, a, \sigma)$ accepts with probability at least $2/3$.

(In words: C verifies that for all x , there exists a y such that $\neg\Phi(x, y)$ is unsatisfiable. In other words, C verifies that for all x , there exists a y such that for all z , we have $\Phi(x, y, z)$. In other words, C verifies the truth of the Π_3^P -sentence S .)

As a point of clarification, whenever we quantify over quantum states (such as σ), we can actually take a tensor product of a polynomial number of copies of the states, as needed. Of course, we can't rule out the possibility that we'll get a state that's entangled across all the registers. Fortunately, though, we don't use the witness state registers in such a way that it ever matters whether they're entangled or not.

As a second point of clarification, in forming the statement above, whenever we have a condition that involves a quantum algorithm (say, V or C) accepting with probability at least $2/3$, it's implied that if the condition fails, then the algorithm accepts with probability at most $1/3$. This makes verifying the condition a quantum polynomial-time operation. Likewise, for part (1), it can be guaranteed that there exists an a such that, for all ρ_1, ρ_2 consistent with a and all Ψ and w , the difference between the two acceptance probabilities is at most (say) $1/20$. In such a case, one can verify in quantum polynomial time that the difference is at most $1/10$.

With these clarifications, it's not hard to see that we've given an $\text{NP}^{\text{NP}^{\text{PromiseQMA}}}$ procedure. The NP at the bottom guesses the classical advice string a . The NP in the middle guesses Ψ for part (2) and x for part (4), and is not needed for parts (1) and (3). Finally, the PromiseQMA on top guesses the quantum advice state σ (or ρ_1, ρ_2 for part (1)), as well as Ψ , w , y , and z as needed. Crucially,

quantum states are only ever guessed in the topmost, PromiseQMA quantifier: once guessed, they never need to be passed on to another quantifier, which is impossible with conventional oracle calls.

But why does the procedure we’ve given correctly decide the Π_3^P -sentence S ? Well, firstly, *if* a is a correct trusted advice string, then part (4) of the procedure just directly expresses S , using the assumed $YQ \cdot QCMA/poly$ algorithm for $coNP$ to eliminate one of the three quantifiers in the usual manner of Karp-Lipton theorems.

That leaves the problem of verifying that a is a correct trusted advice string. Parts (2) and (3) of the procedure verify the latter, by quantifying over all possible SAT instances Ψ of the appropriate polynomial size, and checking that for each one, either Ψ has a satisfying assignment or else there’s a witness w that causes C to accept Φ , but not both. (In other words, C decides $coNP$ in $YQ \cdot QCMA/poly$.)

Now, for parts (2) and (4), we additionally needed an *existential* quantifier over the untrusted quantum advice state σ , which is then verified using the trusted classical advice string a . The reason is that, in parts (2) and (4), the third and final quantifier needed, over the classical strings y , z , or w , happens to be existential—so that third quantifier simply *must* do “double duty” by also guessing the state σ . As mentioned before, passing a quantum state from an earlier quantifier to a later one is impossible with conventional oracle calls.

However, this need to quantify existentially over σ opens up a problem. Namely, what if the existential quantifiers, in parts (2) or (4), can be satisfied by *different* advice states σ —states that are all compatible with a , but that lead to different behaviors of C on some inputs? For example, perhaps there exists an a such that some σ ’s compatible with a give rise to a complete verification procedure for UNSAT, while other σ ’s compatible with a give rise to a sound verification procedure for UNSAT, but the same σ never gives rise to both. If so, then the σ that we find in part (4) need not give rise to a correct $YQ \cdot QCMA/poly$ algorithm for $coNP$.

Fortunately, we can fix this problem using part (1). In part (1), we enforced that *every* state σ compatible with a must give rise to essentially the same behavior on every input. Thus, from that point forward, it doesn’t even matter whether we find σ via a universal quantifier or an existential one: every σ that passes verification will give rise to the same behavior, and parts (2), (3), and (4) are all talking about the same $YQ \cdot QCMA/poly$ procedure that correctly decides $coNP$. \square

References

- [1] Ronald L Rivest, Len Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. 1978. *Foundations of secure computation*, 4(11):169–180.
- [2] Ivan Damgård, Jens Groth, and Gorm Salomonsen. *The Theory and Implementation of an Electronic Voting System*, pages 77–99. Springer US, Boston, MA, 2003.
- [3] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML Confidential: Machine Learning on Encrypted Data. In *Proceedings of the 15th International Conference on Information Security and Cryptology, ICISC’12*, pages 1–21, Berlin, Heidelberg, 2013. Springer-Verlag.
- [4] Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. *On the Relationship between Functional Encryption, Obfuscation, and Fully Homomorphic Encryption*, pages 65–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [5] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS ’13*, pages 40–49, Washington, DC, USA, 2013. IEEE Computer Society.

- [6] Arjan Jeckmans, Andreas Peter, and Pieter Hartel. *Efficient Privacy-Enhanced Familiarity-Based Recommender System*, pages 400–417. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
- [7] Kristin E. Lauter. Practical applications of homomorphic encryption. In *Proceedings of the 2012 ACM Workshop on Cloud Computing Security Workshop, CCSW '12*, pages 57–58, New York, NY, USA, 2012. ACM.
- [8] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC '09*, pages 169–178, New York, NY, USA, 2009. ACM.
- [9] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2014.
- [10] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, October 1997.
- [11] Daniel R. Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26(5):1474–1483, 1997.
- [12] J Niel De Beaudrap, Richard Cleve, John Watrous, et al. Sharp quantum versus classical query complexity separations. *Algorithmica*, 34(4):449–461, 2002.
- [13] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing, STOC '15*, pages 307–316, New York, NY, USA, 2015. ACM.
- [14] Anne Broadbent, Joseph Fitzsimons, and Elham Kashefi. Universal blind quantum computation. In *Proceedings of the 50th Annual Symposium on Foundations of Computer Science, FOCS '09*, pages 517 – 526. IEEE Computer Society, 2009.
- [15] Dorit Aharonov, Michael Ben-Or, and Elad Eban. Interactive proofs for quantum computations. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 453–469, 2010.
- [16] Andrew M. Childs. Secure assisted quantum computation. *Quantum Info. Comput.*, 5(6):456–466, September 2005.
- [17] Alexandru Gheorghiu, Petros Wallden, and Elham Kashefi. Rigidity of quantum steering and one-sided device-independent verifiable quantum computation. *New Journal of Physics*, 19(2):023043, 2017.
- [18] Tomoyuki Morimae and Keisuke Fujii. Blind quantum computation protocol in which alice only makes measurements. *Phys. Rev. A*, 87:050301, May 2013.
- [19] Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G. Rudolph. Efficient universal blind quantum computation. *Phys. Rev. Lett.*, 111:230501, Dec 2013.
- [20] Atul Mantri, Tommaso F. Demarie, and Joseph F. Fitzsimons. Universality of quantum computation with cluster states and (x,y)-plane measurements, 2016. Eprint:arXiv:1607.00758.
- [21] Joseph F. Fitzsimons and Elham Kashefi. Unconditionally verifiable blind computation, 2012. Eprint:arXiv:1203.5217.
- [22] Atul Mantri, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Optimal blind quantum computation. *Phys. Rev. Lett.*, 111:230502, Dec 2013.
- [23] Tomoyuki Morimae, Vedran Dunjko, and Elham Kashefi. Ground state blind quantum computation on aklt state. *Quantum Info. Comput.*, 15(3-4):200–234, March 2015.

- [24] Elham Kashefi and Petros Wallden. Garbled quantum computation, 2016. Eprint:arXiv:1606.06931.
- [25] Elham Kashefi, Luka Music, and Petros Wallden. The quantum cut-and-choose technique and quantum two-party computation, 2017. Eprint:arXiv:1703.03754.
- [26] Anne Broadbent. Delegating private quantum computations. *Canadian Journal of Physics*, 93(9):941–946, 2015.
- [27] Anne Broadbent. How to verify a quantum computation, 2015. Eprint:arXiv:1509.09180.
- [28] Anne Broadbent and Stacey Jeffery. Quantum homomorphic encryption for circuits of low T-gate complexity. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 609–629, 2015.
- [29] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 195–203, New York, NY, USA, 1987. ACM.
- [30] Complexity Zoo. https://complexityzoo.uwaterloo.ca/Complexity_Zoo.
- [31] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 333–342, New York, NY, USA, 2011. ACM.
- [32] Ben W. Reichardt, Falk Unger, and Umesh Vazirani. A classical leash for a quantum system: Command of quantum systems via rigidity of CHSH games. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 321–322, New York, NY, USA, 2013. ACM.
- [33] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. ACM.
- [34] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 97–106, Washington, DC, USA, 2011. IEEE Computer Society.
- [35] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, 2012. ACM.
- [36] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg, 2010. Springer-Verlag.
- [37] Pablo Arrighi and Louis Salvail. Blind quantum computation. *International Journal of Quantum Information*, 04(05):883–898, 2006.
- [38] Yfke Dulek, Christian Schaffner, and Florian Speelman. *Quantum Homomorphic Encryption for Polynomial-Sized Circuits*, pages 3–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
- [39] Joseph F Fitzsimons. Private quantum computation: An introduction to blind quantum computing and related protocols. 2016.

- [40] Gorjan Alagic, Anne Broadbent, Bill Fefferman, Tommaso Gagliardoni, Christian Schaffner, and Michael St. Jules. *Computational Security of Quantum Encryption*, pages 47–71. Springer International Publishing, Cham, 2016.
- [41] Li Yu, Carlos A. Pérez-Delgado, and Joseph F. Fitzsimons. Limitations on information-theoretically-secure quantum homomorphic encryption. *Phys. Rev. A*, 90:050303, Nov 2014.
- [42] Tomoyuki Morimae and Takeshi Koshiha. Impossibility of perfectly-secure delegated quantum computing for classical client, 2014. Eprint:arXiv:1407.1636.
- [43] Vedran Dunjko and Elham Kashefi. Blind quantum computing with two almost identical states, 2016. Eprint:arXiv:1604.01586.
- [44] Atul Mantri, Tommaso F. Demarie, Nicolas C. Menicucci, and Joseph F. Fitzsimons. Flow ambiguity: A path towards classically driven blind quantum computation, 2016. Eprint:arXiv:1608.04633.
- [45] Dan Shepherd and Michael J Bremner. Temporally unstructured quantum computation. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 465, pages 1413–1439. The Royal Society, 2009.
- [46] Matthew McKague. Interactive proofs for BQP via self-tested graph states. *Theory of Computing*, 12(3):1–42, 2016.
- [47] Tomoyuki Morimae and Joseph F. Fitzsimons. Post hoc verification with a single prover, 2016. Eprint:arXiv:1603.06046.
- [48] G. Brassard. A note on the complexity of cryptography (corresp.). *IEEE Transactions on Information Theory*, 25(2):232–233, Mar 1979.
- [49] Leonard Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, SFCS ’78, pages 75–83, Washington, DC, USA, 1978. IEEE Computer Society.
- [50] Robert Raussendorf and Hans J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, May 2001.
- [51] H. J. Briegel, D. E. Browne, W. Dur, R. Raussendorf, and M. Van den Nest. Measurement-based quantum computation. *Nat Phys*, pages 19–26, Jan 2009.
- [52] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pages 113–124. ACM, 2011.
- [53] Chee K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, 26(3):287 – 300, 1983.
- [54] Scott Aaronson. BQP and the polynomial hierarchy. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC ’10, pages 141–150, New York, NY, USA, 2010. ACM.
- [55] Thomas Jansen. On the black-box complexity of example functions: The real jump function. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA ’15, pages 16–24, New York, NY, USA, 2015. ACM.
- [56] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, 41(2):303–332, 1999.

- [57] Michael J Bremner, Richard Jozsa, and Dan J Shepherd. Classical simulation of commuting quantum computations implies collapse of the polynomial hierarchy. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, page rspa20100301. The Royal Society, 2010.
- [58] Scott Aaronson and Andrew Drucker. A full characterization of quantum advice. In *Proceedings of the Forty-second ACM Symposium on Theory of Computing*, STOC '10, pages 131–140, New York, NY, USA, 2010. ACM.
- [59] Adi Shamir. $IP = PSPACE$. *J. ACM*, 39(4):869–877, October 1992.
- [60] The Aaronson \$25.00 prize. <http://www.scottaaronson.com/blog/?p=284>.
- [61] Howard Barnum, Claude Crépeau, Daniel Gottesman, Adam D. Smith, and Alain Tapp. Authentication of quantum messages. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 449–458, 2002.
- [62] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, New York, NY, USA, 10th edition, 2011.
- [63] John Watrous. Quantum computational complexity. In *Encyclopedia of complexity and systems science*, pages 7174–7201. Springer, 2009.
- [64] Scott Aaronson. QMA/qpoly $\not\subseteq$ PSPACE/poly: de-merlinizing quantum protocols. In *in Proceedings of 21st IEEE Conference on Computational Complexity*, 2006.
- [65] Scott Aaronson. The equivalence of sampling and searching. In *Proceedings of the 6th International Conference on Computer Science: Theory and Applications*, CSR'11, pages 1–14, Berlin, Heidelberg, 2011. Springer-Verlag.
- [66] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.
- [67] Clemens Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 17(4):215 – 217, 1983.
- [68] Scott Aaronson. Quantum lower bound for recursive Fourier sampling. *Quantum Info. Comput.*, 3(2):165–174, March 2003.
- [69] J. Watrous. Succinct quantum proofs for properties of finite groups. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 537–, Washington, DC, USA, 2000. IEEE Computer Society.
- [70] Richard M. Karp and Richard J. Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28:191–201, 1982.
- [71] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. In *Computational Complexity, 2007. CCC'07. Twenty-Second Annual IEEE Conference on*, pages 115–128. IEEE, 2007.
- [72] Scott Aaronson and Andrew Drucker. A full characterization of quantum advice, 2010. Eprint:arXiv:1004.0377.